Where am I ?
Contributions to the Localization Problem of Mobile Robots

Von der Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines
Doktor-Ingenieur (Dr.-Ing.)

genehmigte

D i s s e r t a t i o n

von Dipl.-Inform. René Iser
geboren am 18. 07. 1981
in Braunschweig

Eingereicht am: 29. 03. 2012
Mündliche Prüfung am: 14. 09. 2012
1. Referent: Prof. Dr.-Ing. Friedrich M. Wahl
2. Referent: Prof. Dr. rer.-nat. Wolfram Burgard

2012

Danksagung

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Robotik und Prozessinformatik der Technischen Universität Carolo-Wilhelmina zu Braunschweig entstanden.

Bei dem Leiter des Instituts, Herrn Prof. Dr.-Ing. Friedrich M. Wahl bedanke ich mich ganz herzlich für die wertvolle und umfangreiche Unterstützung, sowie die Möglichkeit, sich als Wissenschaftler in einer freien und angenehmen Umgebung entwickeln zu können. Anders wäre die Entstehung dieser Arbeit sicher nicht möglich gewesen. Weiterhin möchte ich mich bei Prof. Dr. rer.-nat. Wolfram Burgard für die Übernahme des Koreferats bedanken. Außerdem bedanke ich mich bei der Deutschen Forschungsgemeinschaft (DFG), welche diese Arbeit teilweise gefördert hat (WA 848/14-3).

Besonderer Dank geht auch an Arthur Martens, der mit viel Entwicklungsarbeit zur Umsetzung der beschriebenen Verfahren beigetragen hat. In diesem Zusammenhang bedanke ich mich bei Simon Winkelbach für die hilfreichen und konstruktiven Vorschläge während der Korrekturphase. Einzelne Teile dieser Arbeit sind in enger Zusammenarbeit mit Daniel Kubus entstanden. Vielen Dank dafür, Daniel. An dieser Stelle bedanke ich mich auch bei Stefan Hößler und Harald Meyer-Kirk ohne deren wertvolle Arbeiten an den Robotern diese Dissertation nicht entstanden wäre.

Insgesamt möchte ich bei allen Mitgliedern des Instituts ganz herzlich für die wunderbaren Jahre bedanken. Dies gilt insbesondere für die kreativen Konversationen in der Teeküche. Ihr seid die besten Kollegen die man sich wünschen kann.

Mein Dank geht auch an Steffen und Chris für die moralische Unterstützung der letzten Jahre. Bei Hannes und Tati möchte ich mich für die wunderbaren Urlaube und Tauchabenteuer bedanken.

Schließlich möchte ich mich bei meiner Babs bedanken. Danke für die schöne Zeit, die wir bisher miteinander hatten und danke, daß du für mich da bist.

Braunschweig, im März 2012 *René Iser*

Kurzfassung

Das Lokalisierungsproblem mobiler Roboter beschreibt die Aufgabe, deren Position und Orientierung, die sog. *Pose*, bezüglich eines gegebenen Weltkoordinatensystems zu bestimmen. Die Fähigkeit zur Selbstlokalisierung wird in vielen Anwendungsbereichen mobiler Roboter benötigt, wie etwa bei dem Materialtransport in der industriellen Fertigung, bei Einsätzen in Katastrophengebieten oder sogar bei der Exploration fremder Planeten. Die genannten Beispiele zeigen die fundamentale Bedeutung der Selbstlokalisierung. Eine Unterteilung existierender Verfahren zur Lösung des genannten Problems erfolgt je nachdem ob eine Lokalisierung auf lokaler oder auf globaler Ebene stattfindet.

Globale Lokalisierungsalgorithmen bestimmen die Pose des Roboter bezüglich eines Weltkoordinatensystems ohne jegliches Vorwissen, wohingegen bei lokalen Verfahren eine grobe Schätzung der Pose vorliegt, z.B. durch gegebene Odometriedaten des Roboters. In Analogie zu einem Roboter, der von seiner aktuellen Pose verschwindet und an einem anderen Ort wieder erscheint, wird das globale Lokalisierungsproblem in der Literatur auch häufig *kidnapped robot* Problem genannt.

Im Rahmen dieser Dissertation wird ein neuer Ansatz zur Lösung des globalen Lokalisierungsproblems vorgestellt. Die grundlegende Idee ist, eine lokale Karte und eine globale Karte in Übereinstimmung zu bringen. Die lokale Karte wird durch ein effizientes Scanmatching-Verfahren generiert. Der beschriebene Ansatz ist äußerst robust sowohl gegenüber Mehrdeutigkeiten der Roboterpose als auch dem Auftreten dynamischer Hindernisse in nicht-statischen Umgebungen. Zur Berechnung der globalen Karte wird ein Verfahren zur Lösung des Simultaneous Localization and Mapping (SLAM) Problems beschrieben, welches darauf basiert, die globale Karte in eine Menge lokaler Kartenfragmente zu unterteilen. Diese werden anschließend mittels eines Optimierungsalgorithmus konsistent aneinandergefügt, welcher auf dem Prinzip der Swarm Intelligence beruht. Der Algorithmus zur globalen Lokalisierung besteht hauptsächlich aus drei Komponenten: Einem Scanmatcher zur Generierung der lokalen Karte, einer Methode zum matchen von lokaler und globaler Karte und einer Instanz, welche entscheidet, wann der Roboter mit hinreichender Sicherheit korrekt lokalisiert ist. Das Matching von lokaler und globaler Karte wird dabei von einer parallelisierten Variante des Random Sample Matching (pRANSAM) durchgeführt, welche eine Menge von Posenhypothesen liefert. Diese Hypothesen werden in einem weiteren Schritt analysiert, um bei hinreichender Eindeutigkeit die korrekte Roboterpose zu bestimmen. Umfangreiche Experimente belegen die Zuverlässigkeit und Genauigkeit des in dieser Dissertation vorgestellten Verfahrens.

Abstract

Self-localization addresses the problem of estimating the position and orientation (the *pose)* of mobile robots with respect to a certain coordinate system of their workspace. It is needed for various mobile robot applications like material handling in industry, disaster zone operations, vacuum cleaning, or even the exploration of foreign planets. Thus, self-localization is a very essential capability. This problem has received considerable attention over the last decades. It can be decomposed into localization on a global and local level.

Global techniques are able to localize the robot without any prior knowledge about its pose with respect to an *a priori* known map. This problem is also referred to as the *kidnapped robot* problem, i.e. a robot disappears from its current pose and is carried to an arbitrary but unknown place.

In contrast, local techniques (*tracking*) aim to correct so-called odometry errors occurring during robot motion. These errors are due to imprecise sensors estimating the active motion parameters like velocity, acceleration, etc.. Hence, tracking is mandatory for mobile robot navigation.

In this thesis, the global localization problem for mobile robots is mainly addressed. The proposed method is based on matching an incremental local map to an *a priori* known global map. This approach is very time and memory efficient and robust to structural ambiguity as well as with respect to the occurrence of dynamic obstacles in non-static environments. Prior to global localization, a map representing the environment is required. To this end, a solution to the SLAM-Problem is proposed which computes a set of local map fragments globally aligned by an optimization routine. Optimization is performed using swarm intelligence. The algorithm for global localization consists of several components like ego motion estimation or global point cloud matching. Nowadays most computers feature multi-core processors and thus map matching is performed by applying a parallelized variant of the Random Sample Matching (pRANSAM) approach originally devised for solving the 3D-puzzle problem. pRANSAM provides a set of hypotheses representing alleged robot poses. Techniques are discussed to postprocess the hypotheses, e.g. to decide when the robot pose is determined with a sufficient accuracy. Furthermore, runtime aspects are considered in order to facilitate localization in real-time. Finally, experimental results demonstrate the robustness of the method proposed in this thesis.

Contents

Chapter 1

Introduction

'*Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities.*'[1]

Mobile robot localization refers to the problem of estimating its position and orientation (the *pose)* with respect to a dedicated world frame. This capability is a fundamental requirement for various application areas like material handling in industry, disaster zone operations, vacuum cleaning, or even the exploration of foreign planets. These exemplary applications have in common that they demand safe navigation which includes path planning and robust path execution. Both tasks are impossible without precise knowledge about the current robot pose. As a consequence, this problem has received considerable attention in the last decades. Solutions proposed in the literature can be classified into localization on a local and global level.

Local techniques aim to *track* the robot which implies that a rough estimate of the pose is available. Thus, so-called odometry errors occurring during robot motion are corrected. These errors originate from noisy sensors providing imprecise motion parameters like velocity, acceleration, etc.. Hence, tracking is mandatory for algorithms dealing with the popular Simultaneous Localization and Mapping (SLAM) - Problem where the aforementioned odometry errors have to be compensated. SLAM refers to the problem of building maps. Sensor readings need to be aligned correctly in order to compensate for odometry errors.

In contrast, given an *a priori* known map, global techniques operate without any prior knowledge about the robot location which substantially increases the complexity of the problem. As stated above, many applications like path planning become impossible if no such estimation is available since every path planning operation requires a dedicated initial pose. In the literature, the global localization problem is also referred to as the *kidnapped robot* problem [2]. Here the robot suddenly disappears from its current location and is taken to an unknown place chosen arbitrarily.

Fig. 1.1 depicting a 2D map of building 079 of the University of Freiburg [1] illustrates the global localization problem. The mobile robot highlighted at the top is equipped

[1]The dataset to build the map can be downloaded from http://radish.sourceforge.net/

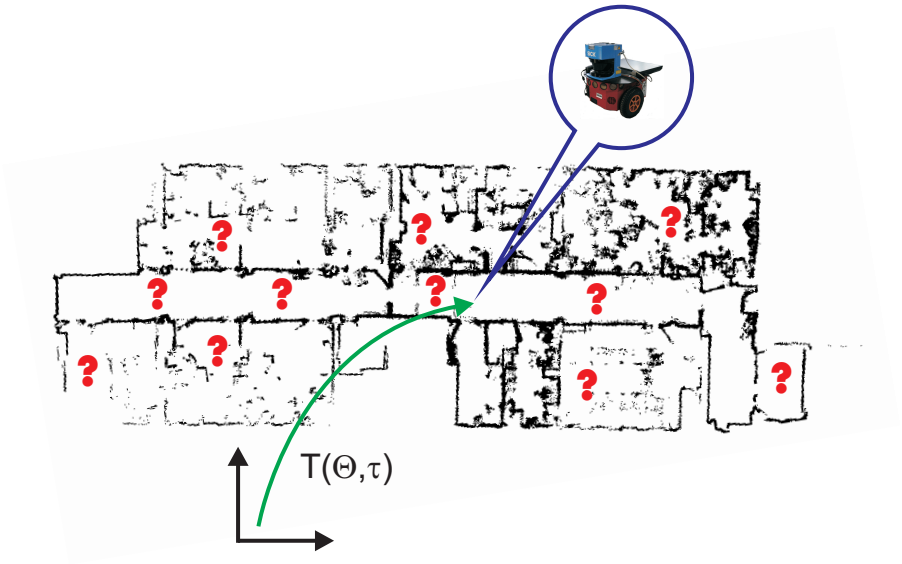
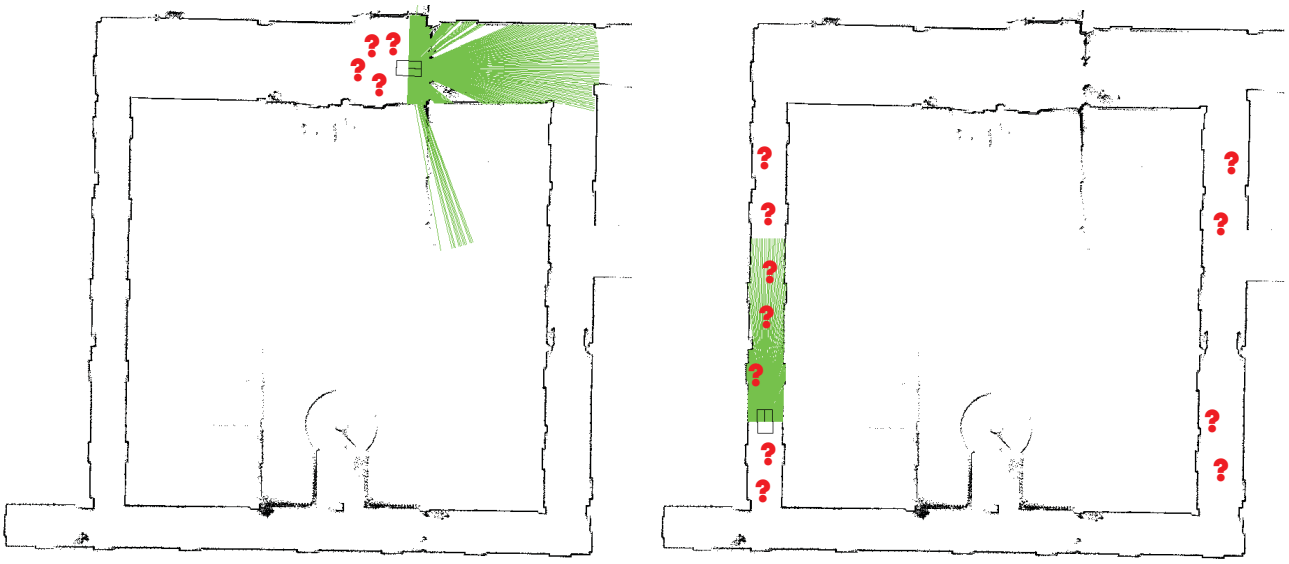


Figure 1.1.: Principle of global localization. A homogeneous transformation $\boldsymbol{T}$ needs to be found representing the correct robot pose with respect to a certain world frame.



(a) (b)

Figure 1.2.: Structural ambiguity might impede the localization process. a) The uncertainty is relatively small since only hypotheses within the local region around the robot are likely to represent the true pose. b) Many hypotheses along both corridors receive a considerable probability of constituting the correct pose.

with a laser range-finder. The question marks distributed over the map indicate the global uncertainty about the robot pose. The objective is to compute a homogeneous transformation ${}^{w}\boldsymbol{T}_{r}(\Theta, \tau)$ with respect to a world frame w which composes a rotation Θ and a translation τ reflecting the true robot pose with a proper accuracy. The overall challenge during global localization is to resolve structural ambiguity, i.e. the robot

pose cannot be uniquely determined using a single sensor reading. Fig. 1.2 illustrates the difficulty caused by structural ambiguity. Suppose the robot is equipped with a laser range-finder. The beams are depicted by the green lines. The uncertainty of the robot pose highlighted in Fig. 1.2a is quite small. Hypotheses outside the local region around the robot are very unlikely. In contrast, the robot pose depicted in Fig. 1.2b entails high uncertainty since hypotheses along both lateral corridors carry non negligible probabilities of representing the true pose. These examples demonstrate that algorithms solving the global localization problem need to gather information over time in order to isolate the correct robot pose. A further issue which needs to be considered refers to dynamic obstacles, e.g. humans normally share the workspace with the robot. Consequently, significant portions of the *known* world may be occluded by unexpected objects causing large discrepancies between real sensor measurements and expected data. As a result, localization is likely to fail if the robot relies on a pure static environment. Other challenges emerge from sensor noise, motion uncertainty, and local modifications of the workspace. The latter aspect is important since furniture is often referred to as *semi-dynamic* obstacles, i.e. they can be removed by humans which strongly changes the appearance of the workspace. In this thesis, a novel localization algorithm is presented capable of dealing with the mentioned imponderabilities. The proposed method exhibits competitive performance compared to standard localization techniques.

1.1. Existing Techniques

Due to the importance of correct knowledge about the global pose of a mobile robot much research focused on this problem in the past. Most of the approaches which exist nowadays rely on different implementations of the Bayes filter (cf. App. A). According to the work of Fox [3] belief representations can be roughly decomposed into Kalman filters, multi-hypothesis approaches, topological approaches, grid-based metric methods, and sample-based techniques. The following review is closely related to the summary of Fox [3].

Kalman filters assume that the belief state can be represented by an unimodal Gaussian distribution, i.e. mean and covariance [4]. The original Kalman filter requires linear system dynamics and observation models. In general this is not the case in robotic applications and thus linearization is typically applied at the current state yielding the Extended Kalman filter (EKF) [2]. EKFs were successfully applied to the localization problem of mobile robots [5, 6, 7, 8]. A key limitation of Kalman filters is the assumption that the initial belief constitutes a unimodal Gaussian, i.e. Kalman filters are therefore not applicable to the global localization problem. Instead, they are proper means to *track* mobile robots during navigation. Limitations of Kalman filter based tracking in comparison with other existing methods were experimentally validated in [9], e.g. bump noise cannot be sufficiently modeled using normal distributions.

On the contrary, multi hypotheses approaches are designed to overcome the limitations

of the Kalman filter concerning the capability of global localization. Here, the belief space is represented as a mixture of Gaussians which allows to maintain multimodal Gaussian probability densities. Multi hypothesis approaches are often referred to as multi hypothesis tracking (MHT) or multi hypothesis localization (MHL) [10]. The integration of several Kalman filters to a mixture of Gaussians proved to solve the global localization problem [10, 11, 12]. The main drawback of this class of Bayes filters is that sophisticated and robust data association techniques are required, i.e. to decide which observed feature is used to update which hypothesis [10]. Furthermore, hypotheses must be added or deleted which inflicts additional sources of failure. Multi hypotheses approaches are typically feature-based localization methods. Consequently, they will fail in the absence of the expected features.

A third category are topological approaches representing the state space of the robot as a discrete set of locations. Here, locations may be doors, corners, rooms, crossings of hallways or even abstract representations [13, 14, 15]. The advantage of topological methods lies in their efficiency since without loss of generality topological structures are not directly correlated to the complexity of the underlying state space. For example, Cummins and Newman [16] localize the robot in the appearance space by processing a discrete set of consecutive camera images. Topological places are defined as feature sets extracted from the images. However, the price of the efficiency is that either only rough metric estimations of the true robot pose are provided or localization is solely performed on a pure topological level [16].

Grid-based metric methods are often referred to as *Markov localization* (ML). The structure of this class of localization algorithms qualifies them to solve the global localization problem [17, 18, 19, 20]. The space of robot poses is represented using fine-grained 3D grids, i.e. two spatial dimensions and one angular dimension. Markov localization is able to approximate arbitrary probability densities. ML iterates over all locations and updates the belief of the individual grid cells. Two very popular examples of real systems using Markov localization are the museum tour guides Rhino and Minerva [21, 22]. Markov localization proved to be very robust to sensor noise. Furthermore, the robot pose can be precisely estimated depending on the resolution of the grid. Simultaneously, the grid resolution entails the disadvantage of the immense computational costs if implemented in a naive manner. Primary modifications for a considerable speed up are pre-computation of the sensor model and a selective update heuristic of the grid cells [19]. The first improvement aims at using a look up table, i.e. the expected sensor distances are stored in the memory which avoids computational expensive ray tracing operations [2]. The latter enhancement updates only grid cells which exceed a defined probability threshold. Consequently, the higher the certainty about the robot pose the less grid cells are updated which allows real time pose tracking. Burgard *et al.* [18] change the resolution of the grid according to the certainty about the robot pose by using a hierarchical data structure similar to octrees. Additionally, the sensor model was subject of enhancement in the past. Fox *et al.* [19] proposed the entropy filter and the distance filter in order to exclude sensor readings caused by dynamic obstacles. For example, humans crossing the sensor's field of view violate the *Markov assumption* (cf.

App. A) which needs to be valid for all Bayes filters [3].

Today sample-based approaches or *particle filters* are the most prominent techniques to achieve global localization. The principle is often referred to as *Monte Carlo localization* (MCL) which is a variant of Markov localization where the belief is represented by a set of samples. In contrast to grid-based methods, MCL is both more efficient and more accurate since the computational resources concentrate on regions in the state space with significant probability. Due to the ability to represent arbitrary probability densities, MCL solves the global localization problem [23, 24, 25, 22]. The algorithm can be summarized to the three steps

- Sampling: Create the particles of the next generation from the proposal distribution, e.g. the motion model of the robot.
- Weighting: Compute a weight for each particle using an appropriate sensor model.
- Resampling: Draw each particle from the new sample set with a probability proportional to its weight.

For further details of the algorithm refer to App. A. Predominantly, two aspects influence the efficiency and the reliability of the filter. The first issue pertains the number of samples since large particle sets substantially increase the computation time. Therefore, sophisticated methods were devised to decrease the number of required particles [26, 3]. Koller and Fratkina [26] determined the sample size based on the likelihood of observations. Fox [3] introduced *kld-sampling* which estimates the required sample size based on the Kullback-Leibler distance. A closed form solution for the number of required particles is derived which assures that the maximum likelihood estimate and the true posterior do not differ more than a defined threshold. Kwok *et al.* [27] introduced real time particle filters where only a smaller subset of all particles is weighted per observation. The posterior is computed using a weighted mixture of the subsets. The weights are chosen by minimizing the Kullback-Leibler distance between mixture-belief and optimal belief. Another problem arises from accurate sensors, e.g. laser range-finders. Given an ideal sensor with zero noise, MCL would fail since all particles which are not located at the true robot pose would obtain zero weights [25, 2]. Therefore, Thrun *et al.* [25] proposed Dual-MCL exchanging the role of proposal distribution and target distribution. Consequently, particles are sampled from the sensor model and weighted according to the motion model. Thus, Dual-MCL works well for highly accurate sensors. Nevertheless, the authors observed that Dual-MCL performs poorly in reality since measurement failures have drastic negative effects on the particle distribution. Thus, Mixture-MCL was additionally introduced combining standard MCL and its dual [25]. A further option to cope with accurate sensors is to add virtual noise to the sensor model. Pfaff *et al.* [28] dynamically adapt the level of noise of the sensor model for each individual particle conditioned on the distance to the nearest particle. The authors observed a higher accuracy and simultaneously decreased the required number of particles. Further improvements compute place-dependent Gaussian mixture models either for individual beams [29] or for entire scans [30].

1.2. Contribution and Organization of this Work

A fifth class of localization paradigms are methods based on the Random-Sample-Consensus (RANSAC). Originally, RANSAC was devised for fitting geometrical primitives into (noisy) point clouds. For mobile robot localization, RANSAC is used to match one or several observations to a global map yielding pose candidates. Compared to the localization methods presented above, RANSAC-based techniques performing accurate metric localization were not intensively discussed in the literature. In particular, publications discussing RANSAC localization in large-scale environments are rare. For a list of relevant literature refer to Chap. 5.2. Predominantly, the advantage of RANSAC localization is that it does not suffer from any filter divergence.

The main contribution of this thesis is to propose a novel localization framework matching local metrical maps to global ones. Fitting the local maps is performed by applying a highly parallelized variant of the Random Sample Matching (RANSAM) approach [31] which yields a set of hypotheses representing possible robot poses. Note that no features are necessary for point cloud matching using RANSAM. Further techniques are devised postprocessing the hypotheses set. Constraints are presented which indicate a successful localization. Furthermore, runtime issues and local map consistency are discussed. To the best of the author's knowledge, existing RANSAC-based localization methods always require feature extraction procedures. Thus, the presented approach is of higher flexibility concerning its application to almost arbitrary environments. Recent publications proposing RANSAC-based methods ignore the fact that localization constraints are essential for the application to real missions. For example, subsequent path planning operations cannot be triggered safely until the robot is certain about its pose with respect to a dedicated world frame.

Prior to the introduction of the localization framework, a robust scan matcher is proposed which facilitates to generate consistent maps of small or even medium size environments. The described scan matcher is used in this thesis to compute the local maps. In addition, an efficient parallelization of RANSAM called pRANSAM is proposed and an optimal thread distribution is derived.

This thesis is organized as follows. Each chapter discusses related works in order to state clearly the according contribution. Moreover, individual experiments are demonstrated for each chapter which clarify the advantages and drawbacks of the described methods. In Chap. 2 the scan matcher based on a particle filter is proposed for the computation of the local maps. This is an essential step since poor scan matching results render an accurate global pose estimation impossible. In order to generate the global map, a more sophisticated SLAM method is introduced in Chap. 3. The proposed approach uses the previously discussed scan matcher to compute a set of local map fragments which are globally connected on a topological level. A consistent map is obtained by an optimization routine based on swarm intelligence. The parallelization of the RANSAM approach is presented in Chap. 4 and interesting characteristics are discussed. In particular, an optimal distribution of the tasks to the processor cores is derived. This method is used in Chap. 5 to match the local map to the global map in order to compute correct pose

candidates. Moreover, the above mentioned localization constraints and other related issues are discussed. Finally, Chap. 6 concludes this thesis.

References

[1] I. J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. on Robotics and Automation*, 7(2):193–204, April 1991.

[2] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. The MIT Press, 2005.

[3] D. Fox. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, 2003.

[4] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[5] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics and Automation*, 7(3):376 –382, June 1991.

[6] J. S. Gutmann, T. Weigel, and B. Nebel. Fast, accurate, and robust self-localization in polygonal environments. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1412 –1419, 1999.

[7] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, pages 167–193, 1990.

[8] K. O. Arras and S. J. Vestli. Hybrid, high-precision localisation for the mail distributing mobile robot system mops. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 4, pages 3129 –3134, May 1998.

[9] J. S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 736 –743, October 1998.

[10] P. Jensfelt and S. Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Trans. on Robotics and Automation*, 17(5):748–760, October 2001.

[11] J. Reuter. Mobile robot self-localization using pdab. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 4, pages 3512 –3518, 2000.

[12] S. I. Roumeliotis and G. A. Bekey. Bayesian estimation and kalman filtering: a unified framework for mobile robot localization. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 3, pages 2985 –2992, 2000.

[13] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 963 –972, November 1996.

[14] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '95)*, pages 1080 – 1087, July 1995.

[15] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Trans. on Robotics and Automation*, 17(2):125 –137, April 2001.

[16] M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[17] K. Konolige and K. Chou. Markov localization using correlation. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 1154–1159, 1999.

[18] W. Burgard, A. Derr, D. Fox, and A. B. Cremers. Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 730 –735, October 1998.

[19] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Artificial Intelligence Research*, 11:391–427, 1999.

[20] C. F. Olson. Probabilistic self-localization for mobile robots. *IEEE Trans. on Robotics and Automation*, 16(1):55 –66, February 2000.

[21] W. Burgard, B. A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1–2):3 – 55, 1999.

[22] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972–999, 2000.

[23] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, 1999.

[24] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99).*, July 1999.

[25] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence Journal*, 128(1-2), 2001.

[26] D. Koller and R. Fratkina. Using learning for approximation in stochastic processes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 287–295. Morgan Kaufmann Publishers Inc., 1998.

[27] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. *Proceedings of the IEEE*, 92(3):469 – 484, March 2004.

[28] P. Pfaff, W. Burgard, and D. Fox. Robust monte-carlo localization using adaptive likelihood models. In *European Robotics Symposium 2006*, volume 22, pages 181–194. Springer-Verlag Berlin Heidelberg, Germany, 2006.

[29] P. Pfaff, C. Plagemann, and W. Burgard. Gaussian mixture models for probabilistic localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 467 –472, May 2008.

[30] P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3539 –3544, September 2008.

[31] S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *Pattern Recognition (DAGM 2006)*, pages 718–728, 2006.

Chapter 2

A Resampling-Based Scan Matching Approach

2.1. Introduction

The capability of estimating its own pose with respect to a certain coordinate system is a fundamental and mandatory requirement of mobile robots. Of course, odometry data are very likely to be available for most of the applications of mobile robots but this data is known to be erroneous and very inaccurate. Errors resulting from false odometry data will accumulate over time. Consequently, the pose uncertainty with respect to a reference frame increases permanently. A typical example is a mobile robot executing a preplanned path. Depending on the environmental conditions it might be of significant importance to stay on this path with a reasonable precision in order to avoid collisions with static or even dynamic obstacles. The key issue is to determine where the robot is located along the path since odometry errors will cause deviations and have to be corrected. To this end, a reliable pose estimation is necessary.
Another application is the exploration of unknown environments. In this case, the robot operates with a complete absence of any map information. Hence, as a first step, a map has to be computed for further missions. Typically, the workspace is observed using onboard sensors, which allow to generate a map by iteratively fusing the sensor information. If the relative motion between two consecutive observations is not known with an adequate precision, the resulting map will be inconsistent and not applicable for further tasks.

One option to correct errors resulting from false or imprecise odometry data is scan matching where sensor information is matched to map information. Map information does not necessarily refer to a complete map of the workspace but it can simply be the last observation or a collection of previous sensor data. Moreover, environmental information can be gathered by many types of sensors e.g. vision, sonar, laser, infrared, etc. [1]. Thus, the self-localization problem is addressed either by metric means (when depth information is available) or by topological means [2]. In this work, primarily laser data are used and thus scan matching is done on a metrical level.

Formally, the scan matching problem can be described as follows. Let $\boldsymbol{P}_{ref}$ be a reference robot pose and let S_{ref} be a reference scan. Subsequently, the robot moves to a new pose $\boldsymbol{P}_{curr}$ and conducts another scan S_{curr} [1]. Then the objective is to find a suitable translation $\boldsymbol{\tau} \in \mathbb{R}^3$ and a rotation $\phi \in \mathcal{SO}(3)$ [1] resulting in a maximum overlap of the scans S_{ref} and S_{curr}. This problem is addressed by minimizing an error function $e : \phi \times \boldsymbol{\tau} \rightarrow \mathbb{R}$. In general, scan matching is performed in 3D but this work assumes the robot moving on a plane. Thus, $\boldsymbol{\tau} \in \mathbb{R}^2$ and ϕ denotes a one dimensional subspace of $\mathcal{SO}(3)$ in the remainder of this chapter. An example of the scan matching problem is illustrated in Fig. 2.1 which depicts scanpoints generated with a Sick LMS 200 laser range-finder.

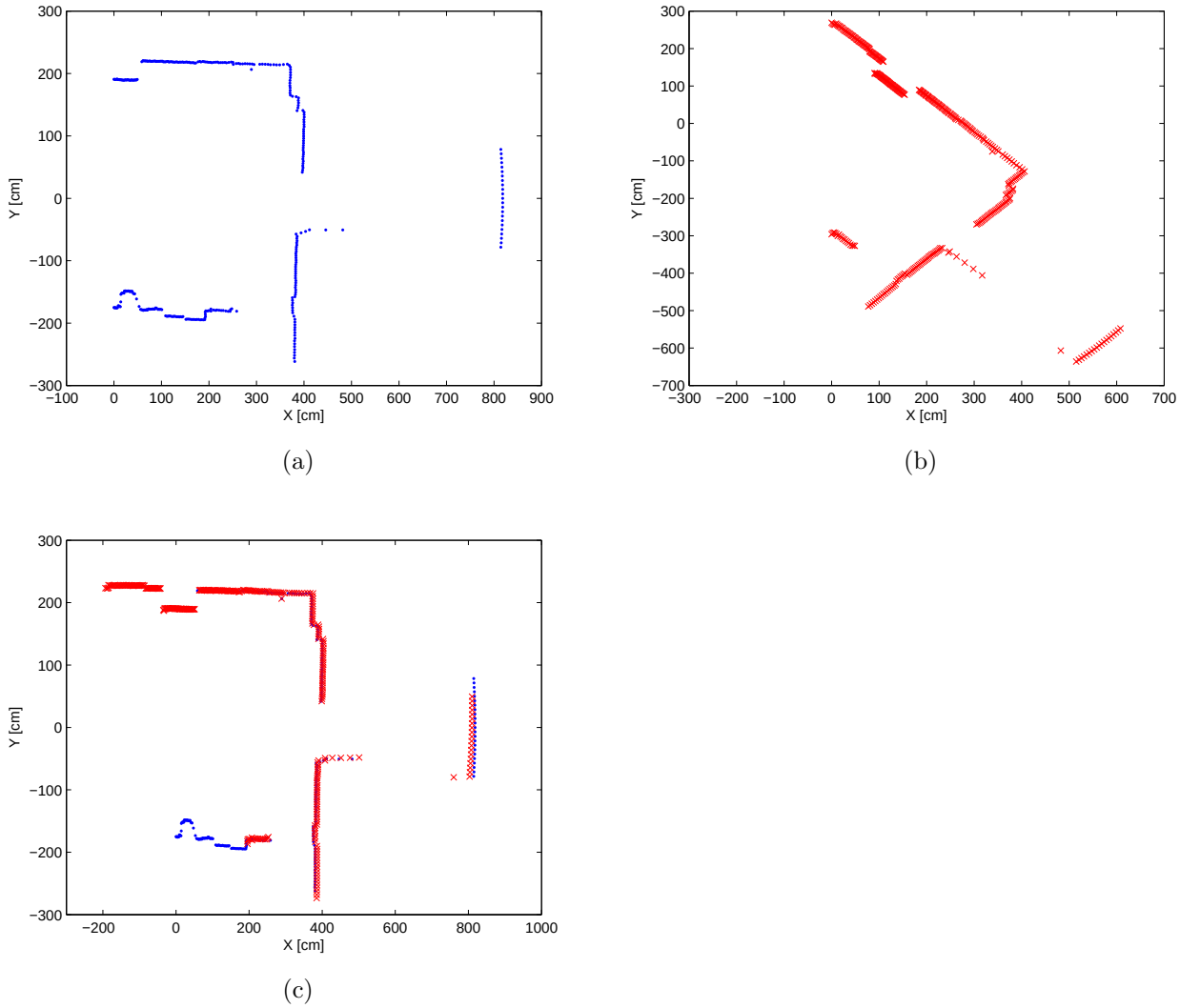


Figure 2.1.: (a) Reference scan S_{ref}. (b) New scan S_{curr} taken at another pose. (c) Matching the new scan S_{curr} to the reference scan S_{ref}.

Fig. 2.1a and 2.1b illustrate the reference scan S_{ref} and second scan S_{curr} taken at a new

[1] A 3 dimensional rotation in the Euclidean space $\mathbb{R}^3$ is a *Lie group* and is often named as $\mathcal{SO}(3)$.

pose. Fig. 2.1c shows both scans aligned by a scan matching process. In this particular case, the correct relative transformation between the scans is $\Delta x = -8$ cm along the x-axis, $\Delta y = 28$ cm in y-direction, and a rotation $\Delta\theta = 43.5°$ around the z-axis of the robot frame perpendicular to the image plane.

The rest of this chapter is organized as follows. First, related works are discussed in Sec. 2.2. In Sec. 2.3 the scan matching approach used throughout this work is described in detail. Experimental results are presented in Sec. 2.4. Finally, Sec. 2.5 concludes this chapter.

2.2. Related Works

A lot of research focused on the scan matching problem during the last decades. All existing algorithms can roughly be categorized into their association method and whether they consider sensor noise or not. One association method is feature to feature matching. For example, features can be lines or corners extracted from laser scans as described by Gutmann [3]. In his work, the method of Cox [4] and the popular Iterative Dual Correspondence (IDC) algorithm proposed by Lu and Milios [1] are combined and a decision rule is derived which of the two algorithms is preferable. The original approach of Cox [4] matches points to lines. Gutmann [3] extended the method to line-to-line feature matching and thus the algorithm is a combination of point-to-point matching (IDC) and feature-to-feature matching. Another feature-to-feature scan matching technique is proposed by Lingemann *et al.* [5] where features are extracted from two consecutive laser scans. More precisely, a feature is an extremum in the polar coordinate system of the sensor which corresponds to corners and jump edges in the Cartesian space.

Except from the point-to-feature scan matcher of Cox [4] as mentioned above, another example for this category of scan matching algorithms is the technique presented by Biber and Strasser [6]. Here a feature is represented by a normal distribution. As a first step, the 2D plane is subdivided into four overlapping regular grids (in order to diminish the effect of discretization). Subsequently, the points of a reference scan are registered at the grid and a normal distribution is computed for each cell representing the probability of a scan point to fall into this cell. For matching a second scan, a corresponding normal distribution is associated to each scan point and the points are weighted according to a score function. Finally, Newton's algorithm is applied to maximize the score.

The most popular class of scan matchers performs point-to-point data association. A well-known method for the registration of point clouds is the Iterative Closest Point (ICP) algorithm proposed by Besl and McKay [7]. ICP requires to compute point-to-point correspondences of two point sets. It iteratively minimizes the mean square error of the point sets until the error changes fall below a predefined threshold. The algorithm always converges to at least a local minimum as proven by the authors. Consequently a lot of enhancements affecting all six stages of the algorithm were proposed over the years, i.e. point selection, matching, weighting of corresponding pairs, rejection of certain pairs, assignment of an error metric, and error minimization. Minguez [8] proposed

a new distance metric for point-to-point correspondences taking the sensor rotation into account. The idea is that points far from the the sensor could be farther away from their corresponding points due to the rotation. A good overview of earlier improvements is provided by Rusinkiewicz [9]. Lu and Milios [1] introduced two well-known approaches, namely the Iterative Matching-Range-Point (IMRP) algorithm and the Iterative Dual Correspondence (IDC) algorithm. In both techniques an error function is minimized which comprises the sum of squared errors of two corresponding sets of points. An important part of this work is the proposal of the so-called Matching-Range-Point rule for correspondence search. To this end, the translation of the point transformation is ignored. For a given data point $\boldsymbol{p}$, the corresponding point $\boldsymbol{p}'$ is computed assuming that the polar angles of both points maximally differ by a defined threshold. Moreover, $|\boldsymbol{p}|$ must be closest to $|\boldsymbol{p}'|$. IDC combines the Matching-Range-Point rule and the standard Closest-Point (CP) rule as employed by ICP. Both rules are combined by applying the matching-range point rule for the rotation search and the CP-rule for the computation of the translation. Sensor uncertainty is not discussed. Montesano *et al.* [10] model scan points as Gaussian random variables. Consequently point-to-point correspondences are computed based on the Mahalanobis distance. In order to calculate the relative transformation of two scans, a least square method minimizes an error function consisting of a sum of squared error terms taking into account the sensor pose uncertainty and the scan point noise, respectively. Jensen and Siegwart [11] also consider pose uncertainty and sensor noise. A new probabilistic distance metric for establishing correspondences of point sets is proposed. Again, a squared error function is minimized by setting the partial derivatives to zero and solving for the pose uncertainty. Diosi and Kleeman [12] introduced a scan matching approach in the polar coordinate system of laser range-finders. The translation is computed by minimizing the sum of weighted range residuals. Orientation correction is based on the assumption that the change of orientation simply is a left or right shift in the polar coordinate system.

Furthermore, many 'alternative' scan matching heuristics were presented in the past. Weiss and Puttkamer [13] introduced a histogram based scan matcher which computes an angle histogram for both scans to correct the rotation. The phase shift is found by calculating the cross-correlation of both histograms. Similarly, the translation is found by computing an x- and y-histogram of the scan points. Afterwards, the phase shift is calculated by cross-correlation again. Censi *et al.* [14] proposed scan matching using the Hough Transform which is computed for both the reference scan and the current scan. Then the scans are matched via cross-correlation. Another correlation based method was presented by Olson [15]; his approach embeds the scan matching problem in a probabilistic framework. The main contribution of his work is to discuss several efficient implementation techniques including a Graphics Processing Units (GPU) implementation.

The technique proposed in this chapter is most comparable to the one of Silver *et al.* [16]. In their work, scan matching is employed in order to track the pose of an Autonomous Underwater Vehicle (AUV) using sonar sensors. For tracking a standard particle filter is used where the particles are weighted by the ICP error. The main difference to

the algorithm presented in this chapter concerns the resampling stage which is only performed once per scan matching operation in their proposed method.

2.3. Resampling-Based Scan Matching

The method explained in this section was published previously [17]. It is called resampling-based scan matching (RBSM) and performs scan matching in 2D. Nevertheless, an extension to 3D is possible. In the following section, a scan refers to the set

$$z_t = \left\{ (r_t^i, \theta_t^i) \mid i = 1..n; n, t \in \mathbb{N}; r, \theta \in \mathbb{R} \right\} \tag{2.1}$$

at time t where r_t^i is a distance measurement (e.g. the range reading of a laser scanner) and θ_t^i is the angle with respect to the sensor frame. Hence, a scan is given in polar coordinates. In this approach, an error function is not minimized in an analytical manner but a set of samples (or *particles*) is spread around the most likely active robot poses and the sample is used which best explains the current sensor reading. Thus, this method operates like a standard particle filter (see App. A.2), but compared to the original algorithm, resampling is performed several times until a convergence criterion is fulfilled leading to a significant improvement concerning the accuracy of the method.

Let

$$\mathcal{X}_t = \left\{ \boldsymbol{x}_t^{[1]}, ..., \boldsymbol{x}_t^{[m]} \mid \boldsymbol{x}_t^{[i]} = \left(x_t^i, y_t^i, \psi_t^i\right) \in \mathbb{R}^2 \times [0, 2\pi], m \in \mathbb{N}, i = 1..m \right\} \tag{2.2}$$

be a set of particles at time t where each particle represents a potential robot pose with respect to a global frame $\mathcal{W}$. Furthermore, the set of scan points

$$\mathcal{S}_{x_t^{[i]}} = \left\{ \boldsymbol{s}^1_{x_t^{[i]}}, ..., \boldsymbol{s}^n_{x_t^{[i]}} \mid \boldsymbol{s}^j_{x_t^{[i]}} \in \mathbb{R}^2, n \in \mathbb{N}, j = 1..n \right\} \tag{2.3}$$

is generated by the particle $\boldsymbol{x}_i^{[i]}$. All points of this set are given with respect to $\mathcal{W}$. A single point $\boldsymbol{s}^j_{x_t^{[i]}}$ always takes the form

$$\boldsymbol{s}^j_{x_t^{[i]}} = \begin{pmatrix} x_t^i \\ y_t^i \end{pmatrix} + r_t^j \begin{pmatrix} cos(\theta_t^j + \psi_t^i) \\ sin(\theta_t^j + \psi_t^i) \end{pmatrix} \tag{2.4}$$

where θ_t^j denotes the local direction of measurement j and $\theta_t^j + \psi_t^i$ is the global direction with respect to $\mathcal{W}$. Fig. 2.2 illustrates the computation of the different sets of scan points. It depicts two different particles with index i and h marked by the red circle and the green circle, respectively. Both particles compute scan points corresponding to measurement direction j and j'. The global positions of the resulting points are different for both particles since their poses with respect to $\mathcal{W}$ are not identical.

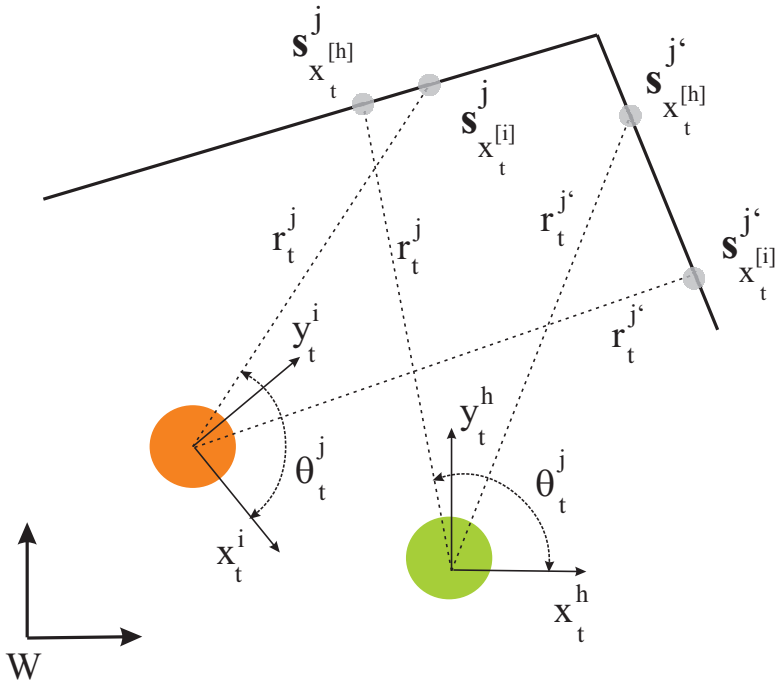


Figure 2.2.: Example of two particles where each particle generates a set of sensor points with respect to a global frame $\mathcal{W}$. In both cases the angle θ denotes the direction of measurement j with respect to the frame of the particle.

As a result the objective is to find a particle $\hat{\boldsymbol{x}}_t$ which is the best approximation for the true current robot pose considering all previous scans and the current odometry data u_t. This means that the particle set $\mathcal{X}_{t-1}$ at time $t-1$ is moved according to a motion model which processes u_t resulting in a new particle set $\widetilde{\mathcal{X}}_t$. When the scan matching routine terminates, $\mathcal{X}_t$ provides $\hat{\boldsymbol{x}}_t$. The scan of the particle $\hat{\boldsymbol{x}}_t$ is stored in a set denoted by $\mathcal{S}_t$, which is formally defined as

$$\mathcal{S}_t = \begin{cases} \mathcal{S}_{\hat{x}_t}, \text{if } t > 0, \\ \mathcal{S}_{\mathcal{O}}, \text{ else} \end{cases} \tag{2.5}$$

where $\mathcal{O}$ is the origin of $\mathcal{W}$. This equation states that the point cloud from measurement at $T = 0$ is computed from a predefined position since no correlation to previous scans is available yet. The point clouds from time $T = 0$ to time $T = t$ yield a set

$$\mathcal{M}_t = \bigcup_{T=0}^{t} \mathcal{S}_T \tag{2.6}$$

which is called the map at time t in the remainder of this section. $\mathcal{M}_t$ is realized as a grid. Thus all points are discretized and stored in their corresponding grid cell. Note the difference between $\mathcal{S}_t$ and $\mathcal{S}_{x_t^{[i]}}$ where the first set contains the points of the best particle, whereas the second set is generated by an arbitrary particle. These data will be

discarded if they are not produced by the best sample. Subsequently, the scan matching process is performed by computing the most likely robot pose with respect to the map at time $t-1$ given z_t:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}_t^{[m]}}{\operatorname{argmax}} \left\{ p(z_t \mid \boldsymbol{x}_t^{[m]}, \mathcal{M}_{t-1}) \right\} \tag{2.7}$$

The idea to find this pose is to resample the particles several times which yields a dense particle concentration around the most promising pose hypotheses. The algorithm can be summarized as follows:

- Move the particles according to the odometry data u_t.
- Sample each particle m from a Gaussian density function with expected value $\boldsymbol{x}_t^{[m]}$ and covariance $\boldsymbol{\Sigma}$. Then weight each particle and resample several times to get a dense distribution around the most likely poses.

In the second step of the algorithm the variance of the Gaussian density function is reduced in every iteration to obtain dense particle clusters. An example of the scan matching method is depicted in Fig. 2.3 which demonstrates the particle evolution.

The current scan is shown in Fig. 2.3a. The initial particle distribution is marked by the stars of Fig. 2.3b. The color encodes the relative weight of the particles. A red star indicates high confidence whereas the green color represents a low weight. The best particle is highlighted by the big red star. The blue dots constitute the map computed previously. Note that the true robot pose is somewhere close to the rightmost particle. Fig. 2.3c shows the particles after the first resampling operation where the particles tend to regions with high probabilities. Nevertheless, the best particle is still located in the middle of mapped area. Fig. 2.3d illustrates the particle set after the third iteration. They are quite densely distributed around the most promising regions. The current best particle has moved to the rightmost cluster which indeed represents the most likely true robot pose. The variance of the weights is reduced significantly which is important for the algorithm to terminate.

A detailed pseudocode description of the scan matching method is given in Algorithm 1 where the input is threefold. First of all, the particle set χ_{t-1} at time $t-1$ is required. Additionally, the current odometry data u_t and the active sensor information z_t is provided.

Line 2 to 4, moves the particles according to the motion model by processing the odometry data u_t.

In Line 5, the so called the effective sample size M_{eff} is set to zero, which is evaluated in each iteration in order to decide on the termination condition. The effective sample size M_{eff} was introduced first by Liu and Jun [18]. It provides a measure of how well a particle set estimates the target distribution (see Appendix A). Grisetti *et al.* [19] employed the effective sample size in order to decide whether resampling is necessary. This is a reliable standard constraint also described in the tutorial of Arumlampalam

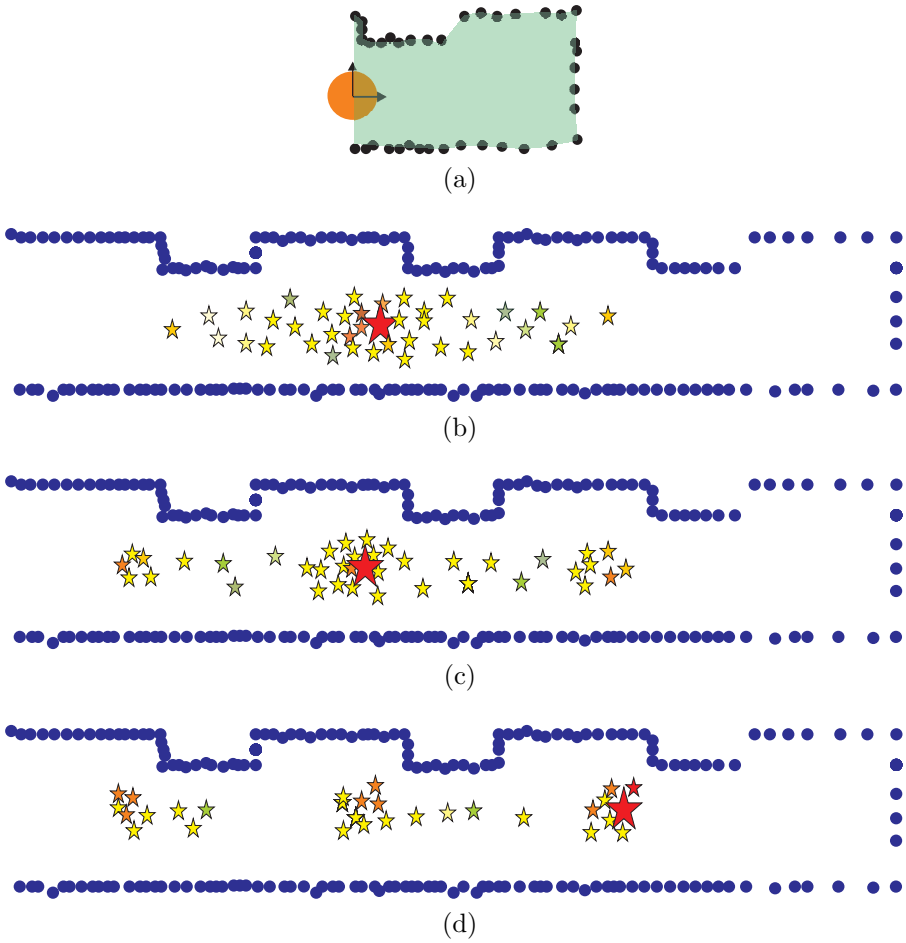


Figure 2.3.: (a) The current scan. The orange circle denotes the robot pose and the green area is covered by the sensor. (b) The initial particle distribution. (c) After one resampling step, the particles begin to push to places which sufficiently explain the current sensor reading. (d) After a few iterations, all particles are densely located around the most likely robot poses. Note that all particles approximately have the same weight.

et al.[20]. Opposed to their method, M_{eff} is used as a decision criterion to determine when resampling needs to be stopped.

Line 6 assigns the particle set χ_{t-1} to the temporary set $\widetilde{\chi}_t$.

Line 7 terminates the algorithm when M_{eff} is greater or equal to $M \cdot \Delta$ with $\Delta < 1$. More precisely, M_{eff} is computed as $\frac{1}{\sum_{i=1}^{M} \left(w_t^{[i]}\right)^2}$ [20] where the sum calculates the variance of the weights. If all particles have the same weight, $M_{eff} = M$ follows, since $w_t^{[i]} = \frac{1}{M}, \forall i \in [1, M]$. In the scan matching application, this is the case if all particles are located in regions with the same probability representing the true robot pose. On the contrary, extensive particle clustering might also yield suboptimal results in environments with sparse features. For example, if the robot operates in a long corridor with bare walls without any corners, the scan matcher is very likely to diverge because the

Algorithm 1 Resampling Scan Matching

1: **procedure** MATCH(χ_{t-1}, u_t, z_t)
2: **for** $i \leftarrow 1$ to M **do**
3: $\boldsymbol{x}_{t-1}^{[i]} = f\left(u_t, \boldsymbol{x}_{t-1}^{[i]}\right)$ ▷ apply the motion model
4: **end for**
5: $M_{eff} = 0$
6: $\widetilde{\chi}_t \leftarrow \chi_{t-1}$
7: **while** $M_{eff} < M \cdot \Delta$ **do**
8: $\overline{\chi}_t \leftarrow \emptyset$
9: **for** $j \leftarrow 1$ to M **do**
10: sample $\boldsymbol{x}_t{}^{[j]} \sim \mathcal{N}(\widetilde{\boldsymbol{x}}_{\boldsymbol{t}}{}^{[j]}, \boldsymbol{\Sigma})$
11: $w_t^{[j]} \leftarrow p(z_t \mid, \boldsymbol{x}_t^{[j]}, \mathcal{M}_{t-1})$
12: *optional: perform a deterministic local correction step and weight again*
13: $\overline{\chi}_t \leftarrow \overline{\chi}_t + < \boldsymbol{x}_t^{[l]}, w_t^{[l]} >$
14: **end for**
15: $\widetilde{\chi}_t \leftarrow \emptyset$
16: **for** $m \leftarrow 1$ to M **do**
17: resample $\boldsymbol{x}_t^{[j]} \propto w_t[j]$
18: $\widetilde{\chi}_t \leftarrow \widetilde{\chi}_t + < \boldsymbol{x}_t^{[j]}, w_t^{[j]} >$
19: **end for**
20: $M_{eff} \leftarrow \frac{1}{\sum_{i=1}^{M} \left(w_t^{[i]}\right)^2}$
21: $\boldsymbol{\Sigma} \leftarrow \delta \cdot \boldsymbol{\Sigma}$
22: **end while**
23: $\chi_t \leftarrow \widetilde{\chi}_t$
24: **return** χ_t
25: **end procedure**

particles will probably be pushed to wrong regions representing local minima. Additionally, the computational burden increases. Thus, depending on the application and the geometrical structure of the environment a well-balanced compromise between a dense particle distribution and the computational effort is essential.

Line 8 initializes another set $\overline{\chi}_t$ as empty.

Line 9 to 14 samples the particles from a Gaussian probability density function with covariance $\boldsymbol{\Sigma}$ in order to push them to regions with high confidence in the following resampling step. Subsequently, a weight is assigned to each particle j given the particle pose and the current map $\mathcal{M}_{t-1}$. The conditional probability is defined as

$$p(z_t \mid, \boldsymbol{x}_t^{[j]}, \mathcal{M}_{t-1}) \propto \sum_{i=1}^{n} \delta_{i,j} \tag{2.8}$$

with

$$\delta_{i,j} = \begin{cases} exp\left\{ -\left(||\boldsymbol{s}^i_{x_t^{[j]}} - \hat{\boldsymbol{m}}|| \right)^2 \right\}, \text{if } ||\boldsymbol{s}^i_{x_t^{[j]}} - \hat{\boldsymbol{m}}|| \leq \xi \\ 0, \text{ else} \end{cases} \tag{2.9}$$

where

$$\hat{\boldsymbol{m}} = \underset{\boldsymbol{m} \in \mathcal{M}_{t-1}}{\operatorname{argmin}} \left\{ ||\boldsymbol{s}^i_{x_t^{[j]}} - \boldsymbol{m}|| \right\} \tag{2.10}$$

A fundamental problem of scan matching is to solve the correspondence problem between an active scan and a previous one, i.e. to find all the points of the environment belonging to both scans. It is problematic to include points of the current scan into the weighting process which do not have a corresponding point in the previous scan since the algorithm tends to fail in this case. Hence, in equation (2.9) an upper bound $\xi \in \mathbb{R}$ is defined which determines the maximum distance between a point of a particle scan and its nearest point in $\mathcal{M}_{t-1}$. Otherwise, the distance is set to zero. Consequently, the nearest neighbor problem can be solved efficiently since only the points belonging to a local grid cell area limited by ξ need to be considered. ξ can be chosen very small if enough samples are involved. Of course, particles with a wrong corresponding relationship between sets of points will always exist, but with a suitable number of particles enough samples approximating the correct correspondences in a sufficient manner will be generated. The optional pose correction step of line 12 is a simple deterministic hill climbing strategy for each particle to find its optimal pose within its local neighborhood. This should decrease the number of resampling steps to obtain acceptable scan matching results. However, although this operation reduces the number of iterations, it might increase the computation time significantly.

Line 15, re-initializes $\widetilde{\chi}_t$ as an empty set because the resampled particles are added to $\widetilde{\chi}_t$.

Resampling is performed in line 16 to 19. Each particle $\boldsymbol{x}_t^{[j]}$ with index j is chosen with a probability proportional to its weight $w_t^{[j]}$.

After computation of M_{eff} (line 20), line 21 decreases the entries of the 3×3 covariance matrix $\boldsymbol{\Sigma}$ by a constant factor $0 < \delta < 1$. This step is very important in order to obtain dense particle clusters. The initial values of the covariance matrix influence the behavior of the algorithm, too. If the initial covariance is large, the particles are widely spread around the expected robot pose. Consequently, the algorithm can deal with a higher odometry uncertainty but more iterations are required until convergence (i.e. $M_{eff} \geq M \cdot \Delta$). The current implementation samples each dimension independently thus the diagonal covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_{xx} & 0 & 0 \\ 0 & \sigma^2_{yy} & 0 \\ 0 & 0 & \sigma^2_{\psi\psi} \end{pmatrix} \tag{2.11}$$

is applied. Another aspect to discuss concerns sensor uncertainty. Currently, no sensor noise is considered which makes the algorithm very hard to combine with quite imprecise sensors like ultrasonic. As mentioned above the approach employs a regular 2D grid and for scan registration the so called end-point model is applied. This means that a grid cell is either occupied or empty and in the case of a laser range-finder, no obstacle occurrence along the beams is regarded. This sort of map representation is called *binary map*. Another option for grid-based map representation are *occupancy grid maps* [21] where each cell contains a probability whether it is occupied or not. Occupancy grid maps are widely used in the field of mobile robotics since they are very flexible concerning the underlying sensor uncertainty model. Thus, occupancy grid maps are predestinated for sensor fusion. An extension of the proposed algorithm which employs occupancy grid maps is possible, but due to the computation of the particle weights, higher computational costs may be inevitable.

(a) (b)

Figure 2.4.: (a) Binary map of the Intel Research Laboratory (b) The corresponding occupancy grid map. This map is taken from Grisetti *et al.* [22].

An example for the difference of binary maps and occupancy grid maps is depicted in Fig. 2.4. Fig. 2.4a shows a simple endpoint model used to represent the map whereas in Fig. 2.4b the more sophisticated occupancy grid map is applied. The darker the regions the higher is the probability of sensing an object. Hence, gray cells mean unexplored regions.

A further option for scan matching under the presence of sensor noise was presented by Burguera *et al.* [23] where two points $\boldsymbol{p}_i \sim \mathcal{N}\left(\hat{\boldsymbol{p}}_i, \boldsymbol{P}_{\boldsymbol{p}_i}\right)$ and $\boldsymbol{q}_j \sim \mathcal{N}\left(\hat{\boldsymbol{q}}_j, \boldsymbol{P}_{\boldsymbol{q}_j}\right)$ from an active scan S_{new} and a reference scan S_{ref} are assumed to be Gaussian random variables. Consequently, the distance between those two points is the squared Mahalanobis distance

$$D^2\left(\boldsymbol{p}_i, \boldsymbol{q}_j\right) = \boldsymbol{h}_{i,j}^T \boldsymbol{C}_{i,j}^{-1} \boldsymbol{h}_{i,j} \tag{2.12}$$

where $\boldsymbol{h}_{i,j} = h\left(\hat{\boldsymbol{x}}_B^A, \boldsymbol{p}_i, \boldsymbol{q}_j\right)$ is the difference between $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$. $\hat{\boldsymbol{x}}_B^A = (x_x, x_y, x_\theta)$ is the expected relative transformation from the new frame A of S_{new} to the reference frame

B of S_{ref}. Now $\boldsymbol{h}_{i,j}$ is computed as

$$\boldsymbol{h}_{i,j} = \begin{bmatrix} x_x + p_x \cdot cos(x_\theta) - p_y \cdot sin(x_\theta) \\ x_y + p_x \cdot sin(x_\theta) + p_y \cdot cos(x_\theta) \end{bmatrix} - \begin{bmatrix} q_x \\ q_y \end{bmatrix}. \tag{2.13}$$

By applying a first order Taylor linearization around $\hat{\boldsymbol{x}}_B^A$,$\hat{\boldsymbol{p}}_i$, and $\hat{\boldsymbol{q}}_j$ the covariance matrix $C_{i,j}$ is computed as

$$\boldsymbol{C}_{i,j} = \boldsymbol{J}_{x_{i,j}} \boldsymbol{P}_B^A \boldsymbol{J}_{x_{i,j}}^T + \boldsymbol{J}_{p_{i,j}} \boldsymbol{P}_{p_i} \boldsymbol{J}_{p_{i,j}}^T + \boldsymbol{P}_{q_j} \tag{2.14}$$

where $\boldsymbol{P}_B^A$ is the covariance of $\hat{\boldsymbol{x}}_B^A$ and

$$\boldsymbol{J}_{x_{i,j}} = \left.\frac{\partial h\left(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}\right)}{\partial \boldsymbol{x}}\right|_{\hat{x}_B^A, \hat{p}_i, \hat{q}_j}, \boldsymbol{J}_{p_{i,j}} = \left.\frac{\partial h\left(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}\right)}{\partial \boldsymbol{p}}\right|_{\hat{x}_B^A, \hat{p}_i, \hat{q}_j}. \tag{2.15}$$

A derivation of the covariance is given in Appendix C. For more details, please consult [23]. Equipped with these equations the weight for each particle j is computed by replacing Eqn. (2.8) by

$$p(z_t \mid, x_t^{[j]}, \mathcal{M}_{t-1}) \propto \sum_{i=1}^{n} e^{-h_{i,j}^T C_{i,j}^{-1} h_{i,j}} \tag{2.16}$$

As a result, the algorithm might properly fit to scan matching applications relying on noisy sensors.

2.4. Experimental Results

The algorithm elucidated in Sec. 2.3 was tested and evaluated on an Intel Core 2 Duo 2 GHz processor with 1024 MB RAM. The RBSM-approach is compared to a trimmed ICP variant [24] and the polar scan matching (PSM) method presented by Diosi and Kleeman [12]. ICP is a self-implementation while the source code of the PSM-algorithm is downloadable from the homepage of the authors[2]. The methods used for comparison were chosen for two reasons. First, ICP is a well-known and very traditional approach and thus it is interesting how the method proposed in this chapter performs compared to ICP. Second, PSM was chosen since it is a relatively new scan matching approach especially designed for laser scan matching. In particular, comparing these methods is very fair because the authors' source code is directly applied to the data used in this study. If not stated otherwise, laser data provided by a SICK LMS 200 laser range-finder was processed. This laser scanner has a field of view of 180° and an angular resolution of 0.5°. In this section, two different kinds of experiments are discussed in order to evaluate the characteristics of the above mentioned techniques. The first experiment is a ground truth experiment where the set-up is inspired by the work of Diosi and Kleeman

[2] http://www.irrc.monash.edu.au/adiosi/downloads.html

[12]. The laser scanner was placed on four known relative poses and Gaussian noise was added to the poses in order to simulate motion uncertainty. Subsequently, sensor data was collected and compared to the result from the different scan matching methods with the ground truth data. This experiment was carried out at four different scenarios. In the second experiment, the scan matchers have to cope with SLAM applications by comparing the subjective quality impression of maps computed by scan matching only. While the first experiment investigates the precision of matching two consecutive scans, the second experiments reveals the effect of the drift which accumulates over time. The data processed in this experiment are either provided by the radish data-set [3] or gathered manually by means of a meccanum wheel omnidirectional drive vehicle.

The algorithms were configured as follows. The translational elements of the initial covariance matrix of Eqn. 2.13 of the RBSM-approach were set to $\sigma_{xx}^2 = \sigma_{yy}^2 = 4\ pix^2$ and the rotational part was set to $\sigma_{33} = 3^{\circ 2}$. Here, pix refers to a grid cell. Consequently, the translational components depend on the resolution of the grid. It could be shown by empirical investigations, that these values perform well in typical scan matching scenarios. Furthermore, Δ was set to $\frac{4}{5}$ (cf. Alg. 1, line 7). The upper bound ξ to determine the maximum distance between two points was set to $\sqrt{2}$. Consequently, the algorithm only searches the adjacent grid cells around the endpoint of a laser beam for corresponding points.

The convergence criterion of ICP was set to $\varepsilon < 10^{-3}$ which means that the difference of the least squares error of two consecutive iterations must be smaller than 10^{-3}. Additionally, the 80 % best corresponding points were chosen for the calculation of the relative transformation.

The configuration parameters of PSM were selected according to the recommendation of Diosi and Kleeman [12]. Table 2.1 depicts the parameters and its values. For more details concerning the meaning of the variables refer to [12]. PSM terminates if either $\varepsilon <$ PM_STOP_COND or if the number of iterations has exceeded PM_MAX_ITER.

Parameter	Value
PM_MAX_ERROR	100 cm
PM_MAX_RANGE	800 cm
PM_MAX_ITER	30
PM_MIN_VALID_POINTS	40
PM_MAX_DIFF	20 cm
PM_SEARCH_WINDOW	20°
PM_STOP_COND	0.4
PM_MEDIAN_WINDOW	5

Table 2.1.: Configuration of PSM

[3] http://radish.sourceforge.net/

2.4.1. Ground Truth Evaluation

In order to evaluate the precision of the RBSM method compared to the other approaches, the SICK laser was placed on a plastic sheet with dimensions 841 mm × 1189 mm. Several outlines of the laser are drawn on different positions on the sheet. The laser is mounted on a rack. Its orientation is adjustable in 15° steps.

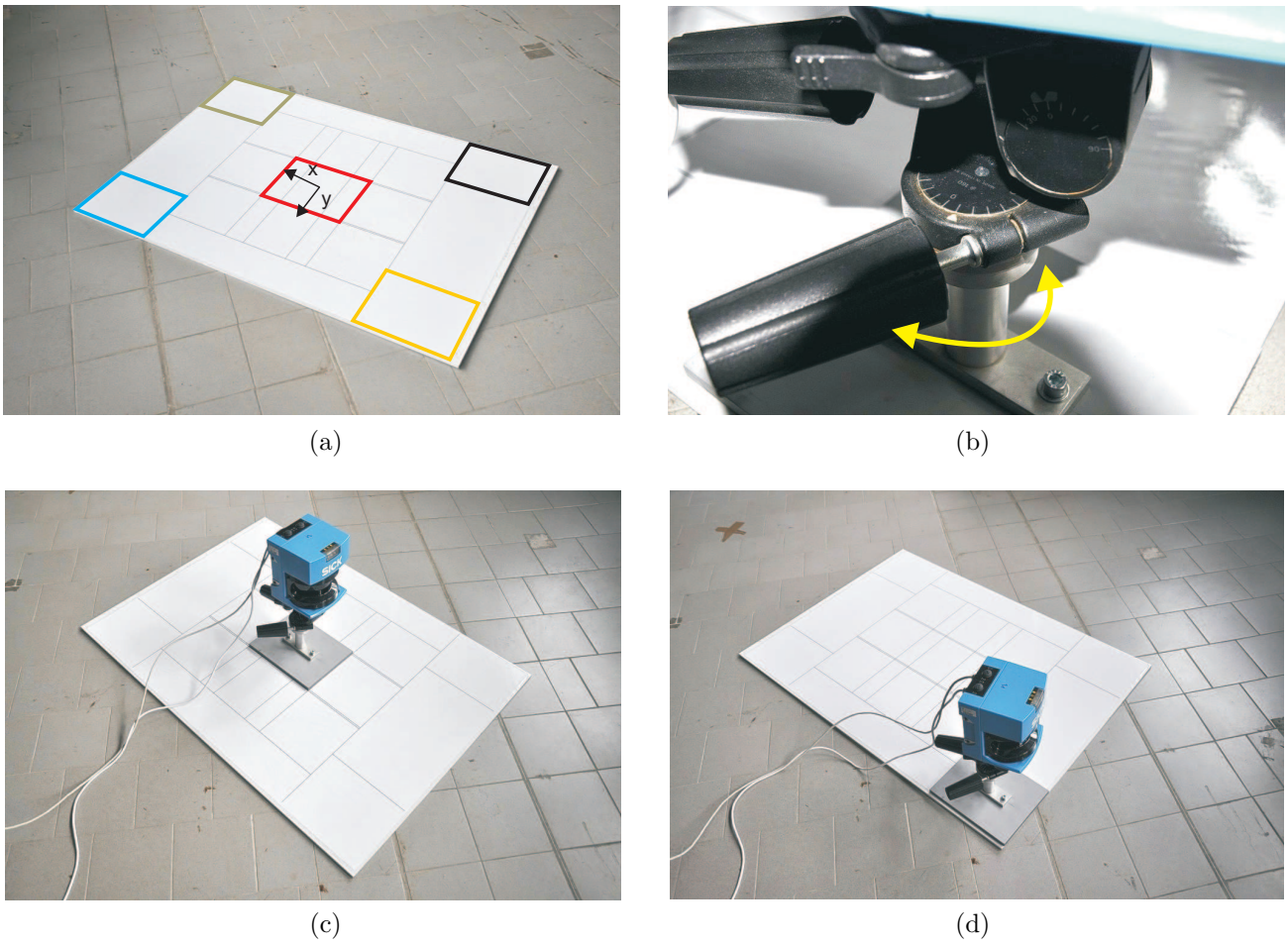
No.	x [cm]	y [cm]	ψ [°]
1	-46.3 cm	-31.8 cm	0°
2	46.3 cm	-31.8 cm	15°
3	46.3 cm	31.8 cm	30°
4	-46.3 cm	31.8 cm	45°

Table 2.2.: Poses of the SICK laser with respect to the middle outline at the four outer corners of the plastic sheet.

An overview of the position and orientation of the laser with respect to the middle of the plastic sheet is given in table 2.2. The laser was positioned on the corners of the sheet with different orientations. Thus, five laser scans were conducted per scenario (one reference scan and four scans employed for matching). The relative displacement of the laser represents odometry data. Hence, Gaussian noise with translational standard deviations $\sigma_{xx} = \sigma_{yy} = 10$ cm and a rotational standard deviation $\sigma_{\psi\psi} = 6°$ was added to the ground truth. As mentioned above, the behavior of the algorithms was investigated in four different scenarios. The grid resolution of RBSM was set to 1 cm. The experimental setup is depicted in Fig. 2.5. Fig. 2.5a shows the plastic sheet and highlights the outlines for the laser positions. The unit to adjust the orientation can be seen in Fig. 2.5b while Fig. 2.5c and Fig. 2.5d depict the reference position and the position of match 2, respectively. Since RBSM is a probabilistic algorithm the results presented in this section are average results over 100 scan matching operations.

Scenario 1 represents a very cluttered environment where the performance of all three algorithms is excellent. The error vectors are depicted in table 2.3. The rightmost column shows the initial pose of each match number. As can be seen, only the ICP result of match number 4 yields significant errors in x- and y-direction. This result is probably due to wrong point to point associations. The matched points of match 2 and match 4 are exemplary illustrated in Fig. 2.6a to Fig. 2.6f. 2.6f clearly shows the drift of the ICP matching result of match 4. The orientation was always found with a proper accuracy since the maximal error is 1.35° (RBSM, match 3).

Scenario 2 represents a simple structured environment. The error vectors are depicted in table 2.4 where the rightmost column again indicates the initial pose of each match number. RBSM exhibits a significant translational drift at match 1 which was expected since the initial error (right column) is considerable. Again PSM shows good results for all matches while the ICP error is large for match 4. The matched points of match 1 and match 3 are exemplary illustrated in Fig. 2.7a to Fig. 2.7f. Also from an optical point of view it is obvious that PSM and ICP outperform RBSM in match 1 and 3. Fig.



(a) (b)

(c) (d)

Figure 2.5.: a) The colors encode the match numbers: black - match 1, green - match 2, blue - match 3, yellow - match 4 b) The unit to adjust the orientation of the laser c) The reference position d) Position of match 2 with 15° orientation

No.	RBSM	ICP	PSM	Start
1	(0.36, 0.39, 0.11)	(0.19, 3e-4, 0.017)	(1.9, 0.42, 0.021)	(-38.0, -38.8, 6.27)
2	(0.58, 0.29, 0.25)	(8.73, 2.13, 0.12)	(0.85, 0.42, 0.51)	(41.9, -21.1, 14.2)
3	(1.63, 3.7, 1.35)	(1.8, 0.55, 1.05)	(2.3, 0.39, 0.17)	(42.8, 18.7, 30.6)
4	(0.7, 3.4, 0.18)	(72.5, 23.4 , 1.05)	(2.7, 0.06, 0.1)	(-49.7, 25.7, 45.7)

Table 2.3.: Scan matching error for all four scans of the first scenario in x[cm],y[cm],ψ[deg]. The rightmost column shows the initial pose.

2.7a clearly highlights the suboptimal matching result of RBSM.

The data of scenario 3 were gathered in a corridor environment. The error vectors are depicted in table 2.5. In this example, ICP exhibits the largest distance to ground truth for match 1 and 4 since a significant deviation in x-direction of 42 cm and 57 cm can be observed. PSM exhibits a poor result for match 4 because the deviation is 16 cm in y-direction. The results of RBSM are of acceptable quality. The error in x-direction

No.	RBSM	ICP	PSM	Start
1	(14.05, 4.7, 1.73)	(0.26, 1.05, 0.15)	(3.98, 1.78, 1.24)	(-62.2, -14.3, -4.3)
2	(1.01, 1.1, 0.47)	(4.75, 0.72, 0.3)	(0.18, 1.2, 0.07)	(48.6, -18.24, 19.3)
3	(6.28, 5.15, 0.73)	(0.71, 1.5, 0.36)	(0.53, 0.08, 0.19)	(35.0, 19.9, 22.8)
4	(1.38, 5.3, 0.5)	(13.9, 11.7 , 3.76)	(6.3, 0.72, 0.78)	(-39.7, 25.8, 41.5)

Table 2.4.: Scan matching error for all four scans of the second scenario in x[cm],y[cm],ψ[deg]. The rightmost column shows the initial pose.

is considerable on average which is normal in a corridor environment. The reason is that only very few corners determine the correct overlap of the point sets. Thus, the particles are very likely to diverge to regions causing a suboptimal matching result. The correction of the initial orientation error is excellent since no orientation differs more than 0.8° from ground truth after the matching operation. The point sets of match 2 and 3 are shown in Fig. 2.8a to 2.8f. The results are of comparable quality.

No.	RBSM	ICP	PSM	Start
1	(8.73, 1.3, 0.44)	(42.2, 0.12, 0.04)	(0.81, 4.6, 0.11)	(-57.9, -21.8, -5.8)
2	(3.89, 0.76, 0.8)	(0.43, 0.88, 0.61)	(0.9, 6.2, 0.47)	(44.6, -45.6, 16.0)
3	(6.25, 1.65, 0.37)	(5.02, 1.81, 0.34)	(0.65, 2.51, 0.19)	(41.1, 44.0, 34.3)
4	(2.73, 6.48, 0.5)	(56.9, 4.20 , 0.34)	(4.18, 15.9, 0.07)	(-56.5, 35.9, 40.4)

Table 2.5.: Scan matching error for all four scans of the third scenario in x[cm],y[cm],ψ[deg]. The rightmost column shows the initial pose.

Scenario 4 represents the same area as scenario 2 from a different point of view. The error vectors are depicted in table 2.6. In this scenario, RBSM clearly outperforms ICP and PSM. ICP exhibits a large rotational error of 10.35° at match 1. Furthermore, a significant drift of 32.9 cm from ground truth in x-direction can be observed. However, the result of match 2, 3, and 4 are good. PSM yields large rotational errors of 6.35° and 4.5° at match 2 and 3. Additionally, the positional error was not properly corrected. Match 2 has a 16.34° deviation from ground truth in x-direction. Match 3 yields a 9.55 cm and 9.15 cm error in x- and y-direction. Opposed to these results, RBSM always causes moderate alignment errors. The results for match 2 and 3 are displayed in Fig. 2.9a to 2.9f. Note the immense rotational error caused by PSM in Fig. 2.9c and 2.9d.

No.	RBSM	ICP	PSM	Start
1	(2.5, 1.3, 0.88)	(32.9, 7.3, 10.35)	(0.39, 0.01, 0.55)	(-51.1, -22.0, 10.43)
2	(2.44, 1.63, 0.73)	(2.67, 0.70, 0.27)	(16.34, 3.4, 6.35)	(43.6, -44.8, 15.4)
3	(0.37, 2.20, 0.14)	(0.22, 2.38, 0.02)	(9.55, 9.15, 4.5)	(48.6, 38.4, 26.7)
4	(1.7, 7.02, 1.30)	(0.95, 6.34 , 1.04)	(5.33, 0.19, 0.08)	(-46.4, 20.8, 36.3)

Table 2.6.: Scan matching error for all four scans of the fourth scenario in x[cm],y[cm],ψ[deg]. The rightmost column shows the initial pose.

Another aspect which needs to be investigated is the dependency of the number of par-

ticles employed by RBSM, the matching error, and the run time. Fig. 2.10a displays the translational error of scenario 1 plotted against the number of particles. The rotational error is depicted in Fig. 2.10b. As can be seen, the curves converge when approximately 1000 particles were applied. Thus, 1000 particles were used in all scenarios discussed above because no example was found where more particles yield a considerable better matching result. Fig. 2.10c and 2.10d show the error evolution of scenario 2. In Fig. 2.10e the run-time is plotted against the number of particles. RBSM configured with 1000 particles normally requires around 0.16 seconds to compute the best overlap. On the contrary, ICP only needs 0.03 seconds computation time and PSM even terminates in 0.002 seconds. Thus RBSM is one and two orders of magnitude slower than ICP and PSM, respectively. However, RBSM clearly outperformed ICP and PSM with respect to average accuracy since it did not cause such remarkable outliers as ICP and PSM.

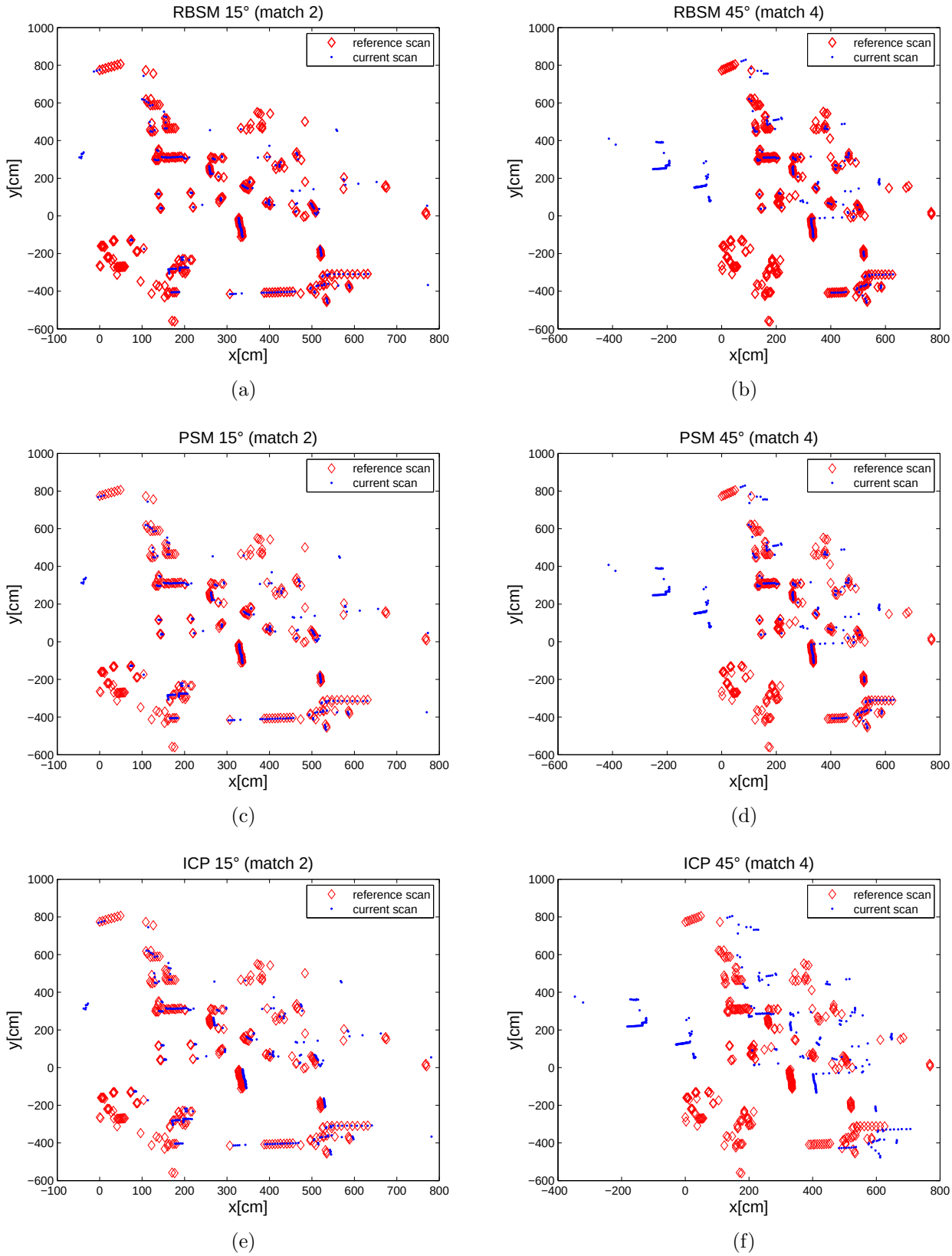


Figure 2.6.: Matching result of the first scenario.

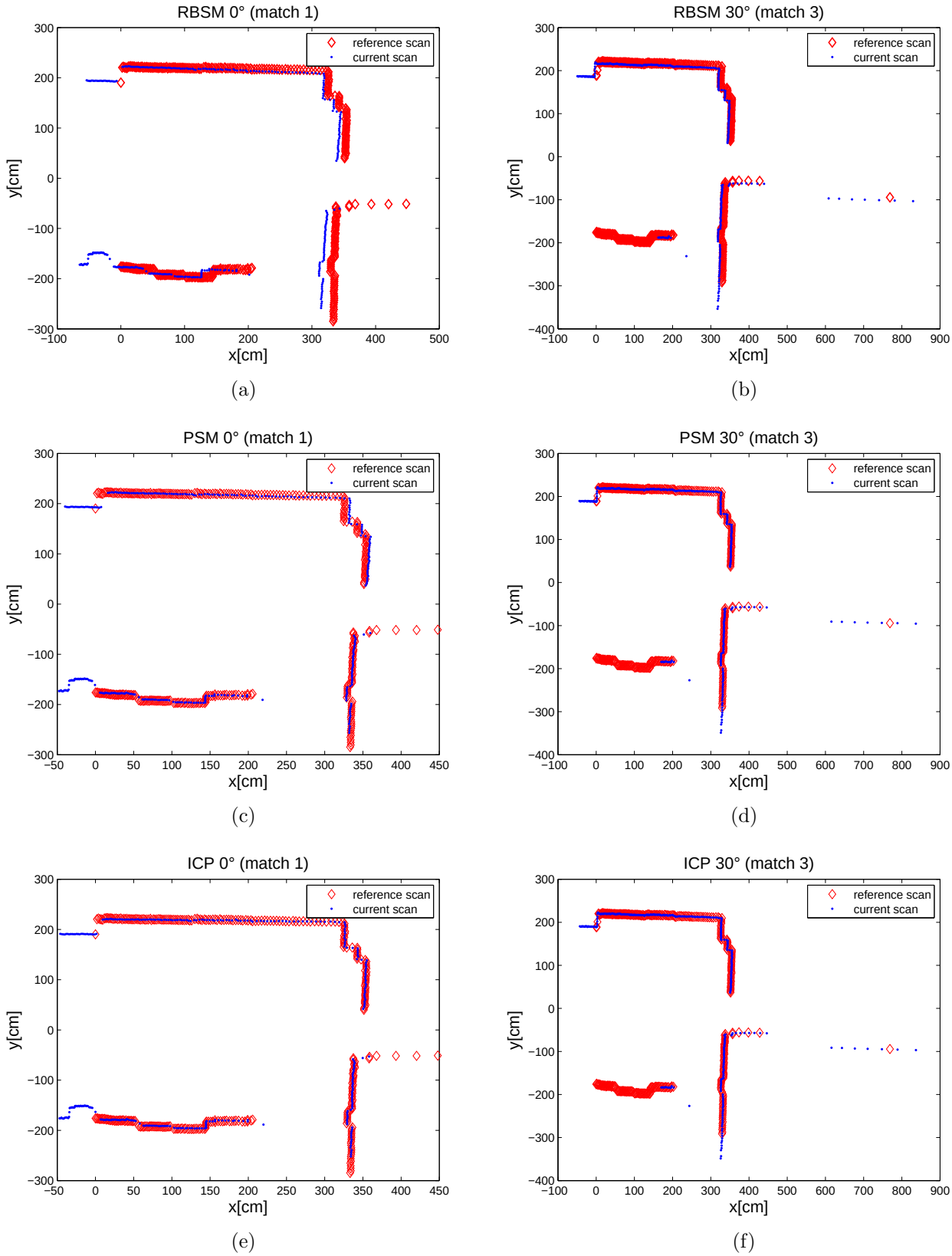


Figure 2.7.: Matching result of the second scenario.

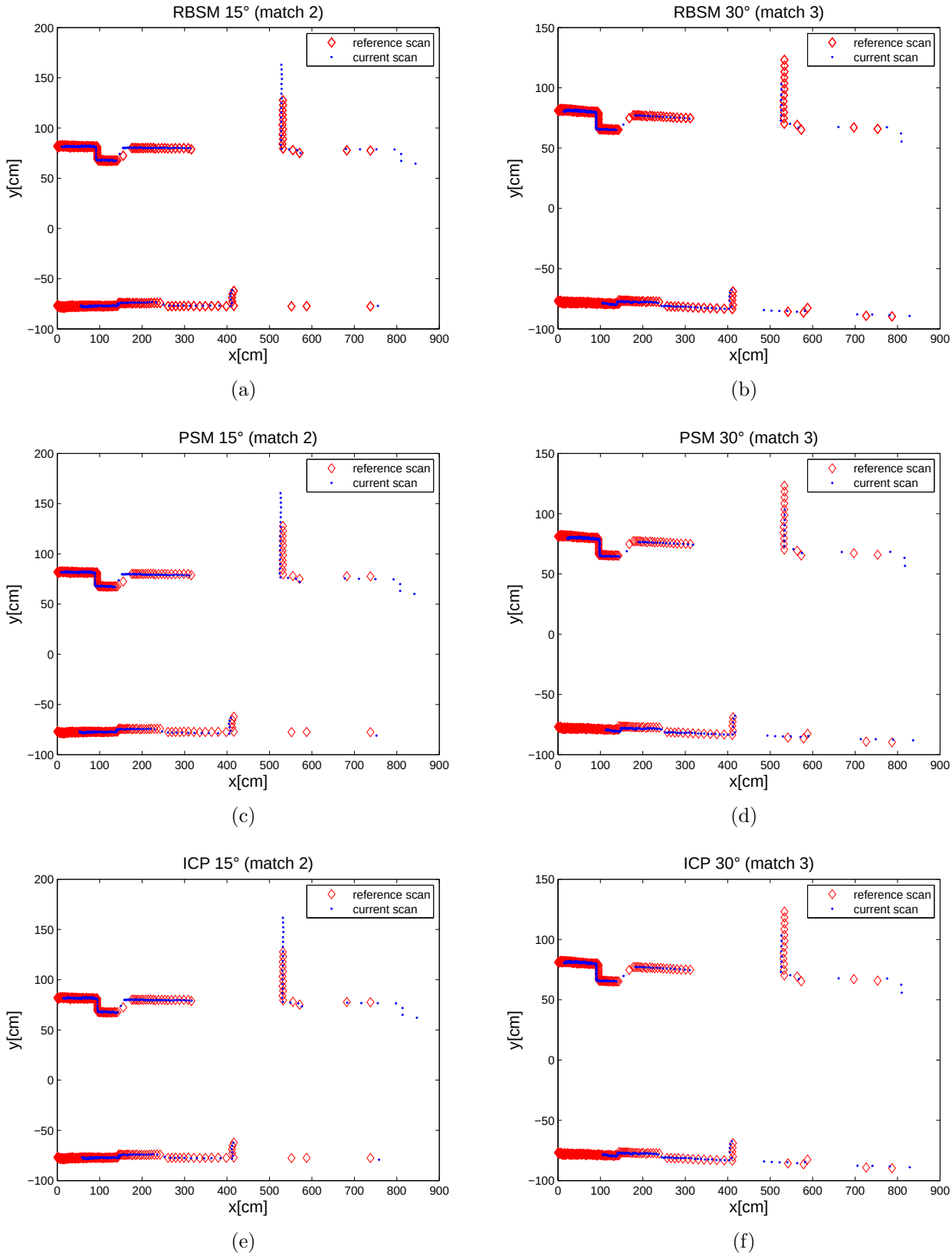


Figure 2.8.: Matching result of the third scenario.

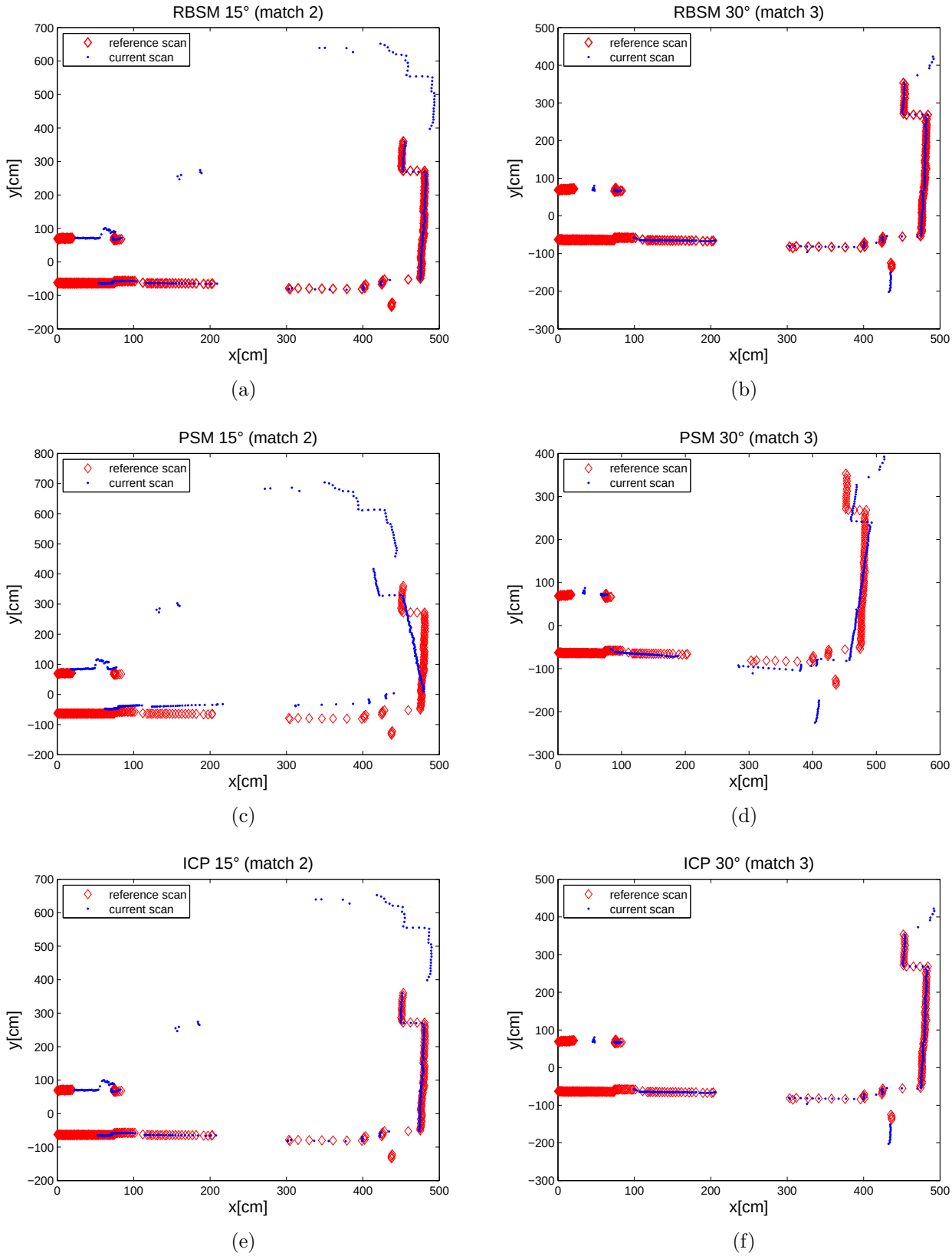
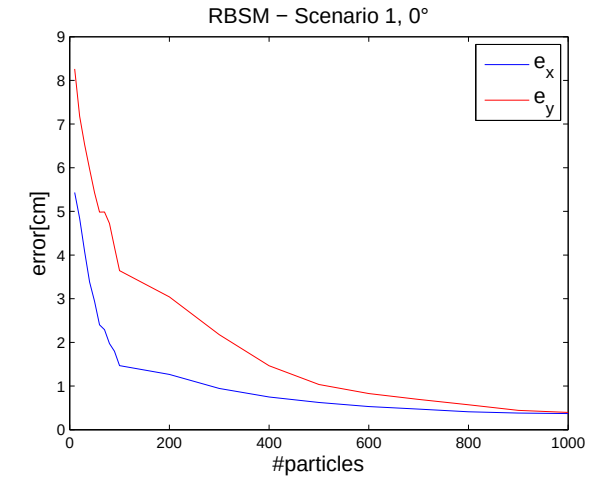
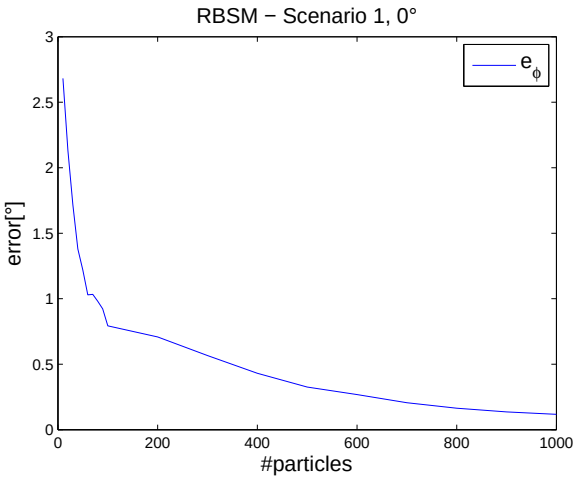


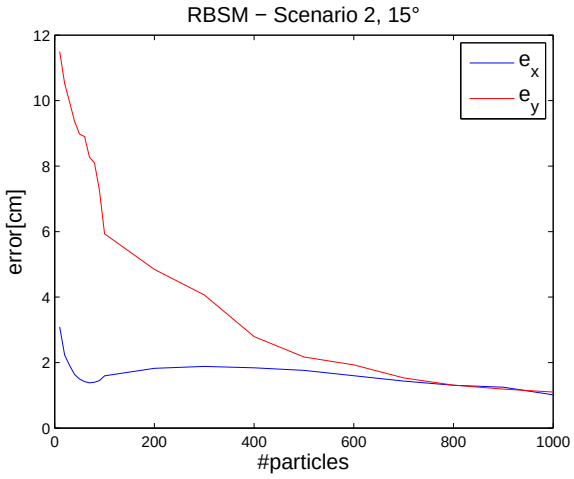
Figure 2.9.: Matching result of the fourth scenario.



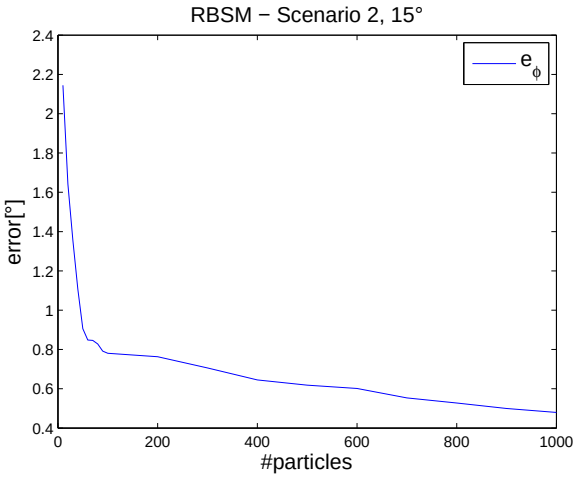
(a)



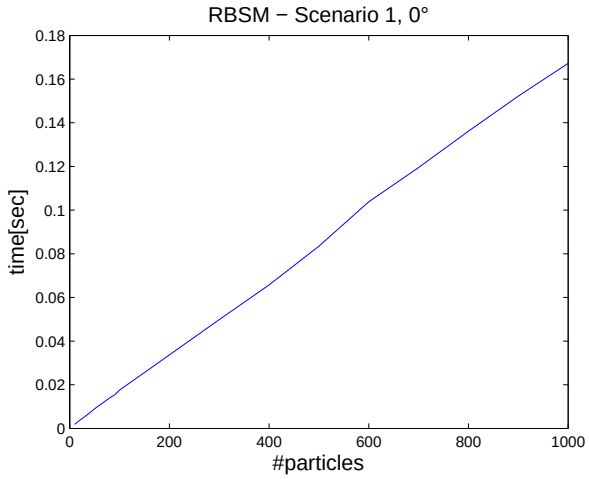
(b)



(c)



(d)



(e)

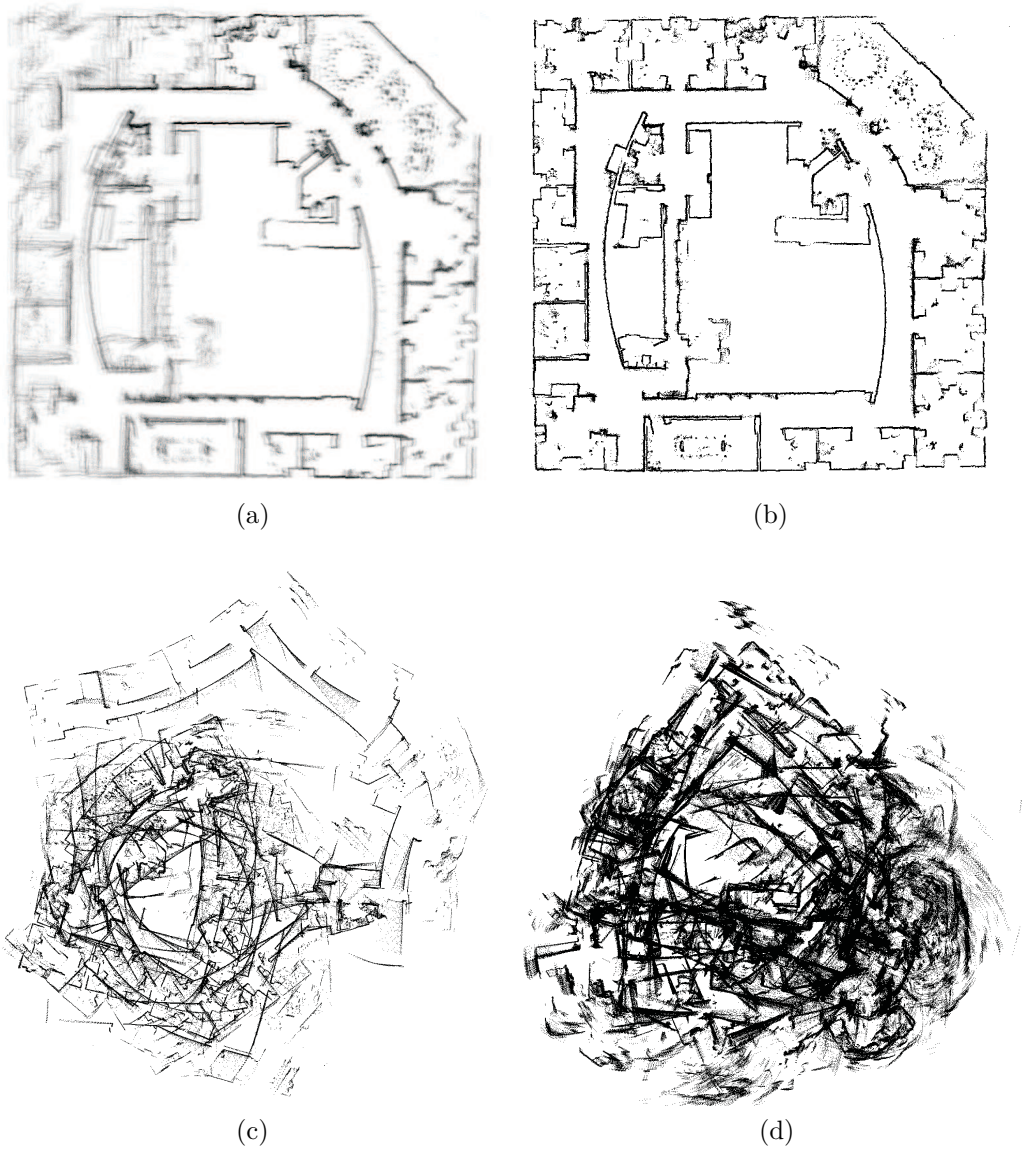
Figure 2.10.: Evolution of the scan matching error of the first and second scenario and the matching time plotted against the number of particles.

2.4.2. Map Building

In this experiment, the scan matching algorithms are applied to SLAM data sets meaning that the scan matchers build maps from many consecutive laser range-readings. RBSM, ICP, and PSM are compared by processing four different data sets. Then the subjective map quality of each scenario and scan matching technique is evaluated. 1000 particles were used with RBSM for map building. Depending on the geometrical structure of the environment, less particles might be sufficient but rather complex scenarios were investigated, i.e. each scenario contains at least one loop and the scan matchers have to keep the map consistent. Hence, 1000 particles are a good compromise between accuracy and computational burden.

The first data set was recorded at the *Intel Research Laboratory* in Seattle and can be downloaded from the Radish Data Set Repository (cf. Sec. 2.4). The laboratory has a dimension of about 28 m × 28 m. The SICK laser data were recorded using a Pioneer II robot [25]. The beam resolution was set to 1°. The map computed by the RBSM approach is depicted in Fig. 2.11a and 2.11b. The best map which could be obtained is displayed in Fig. 2.11b. The scan matcher was able to close two loops. The first loop is the outer one and the second loop includes the corridor connecting the upper and lower part of the map. Since this scan matcher is based on random, stability is an important criterion for the usefulness of the proposed method. Therefore, Fig. 2.11a shows the *average* map generated over 20 trials. Although the average map is blurred in the left half, this image indicates a low variance of the solution. Consequently, all solutions are of comparable quality. In contrast, Fig. 2.11c and 2.11d depict the maps provided by ICP and PSM; Fig. 2.11c shows the ICP result while Fig. 2.11d illustrates the map computed by PSM. Both approaches completely diverged over time because several poor matching results accumulated the errors very fast. A considerable number of bad orientation alignments for both methods were observed which has a strong influence on the map quality. This is probably due to the log file containing many poor odometry estimates which could be compensated by the RBSM method, but forced ICP and PSM to get stuck in local minima frequently. All maps have a resolution of 4 cm.

The second data set was recorded in the first floor of the robotics laboratory of the Institut für Robotik und Prozessinformatik (iRP, Technische Universität Braunschweig). The area has a size of about 22 m × 24 m. The aforementioned meccanum wheel omnidirectional drive vehicle was used in order to gather the sensor data. A scan was triggered whenever the robot conducted a translational motion of 20 cm or after a rotation of 20°. Fig. 2.12b shows a map with 3 cm resolution generated by the RBSM method. It is illustrated that the environment is of rectangular shape consisting of four long corridors which lowers the probability of the scan matcher to keep the map consistent. The circular object at the bottom represents a spiral stair. The time required to collect the data was about 40 minutes but only 72 seconds were required to compute the map. This corresponds to a frequency of about 6 Hz, i.e. RBSM is fast enough to process the data *online*. Fig. 2.12a depicts the average map over 20 trials. Again the mean quality is excellent which implies a low variance. The results of ICP and PSM



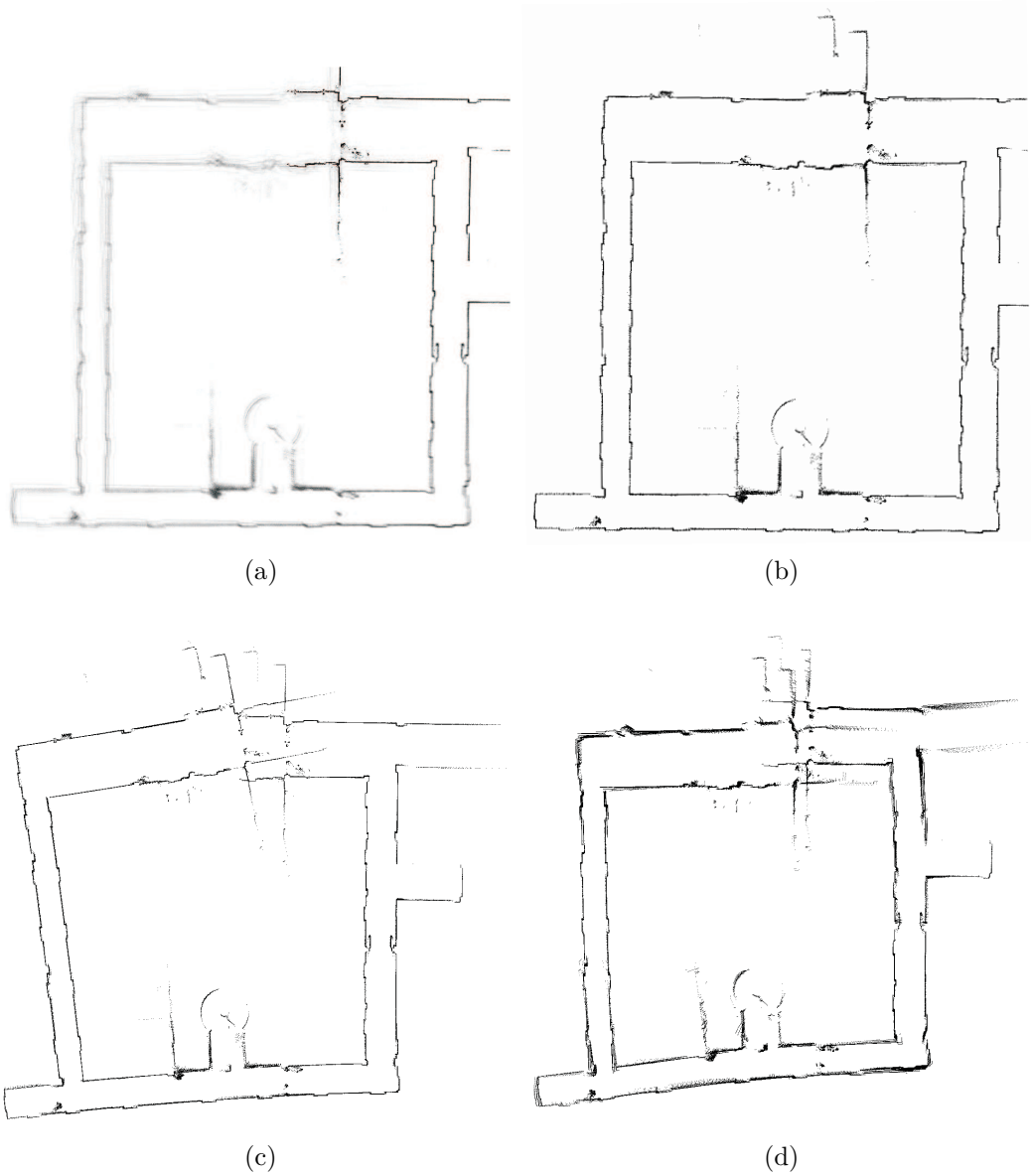
(a) (b)

(c) (d)

Figure 2.11.: Maps of the Intel Research Laboratory with 4 cm resolution. a) RBSM average map computed over 20 iterations. b) Best result of RBSM c) Result provided by ICP d) Result provided by PSM

are depicted in Fig. 2.12c and 2.12d, respectively. The accuracy of the maps are much better compared to the intel data set. Nevertheless, in both cases loop closing was not successful. The corridors of the ICP map are slightly curved but these errors suffice to yield an inconsistent map at the loop closing point. By looking at the maps, the overall errors induced by PSM seem to be larger compared to ICP. However, PSM is closer to the true loop closing pose than ICP for both the translational and rotational component.

The third data set was recorded at building 079 of the AIS laboratory of the University of Freiburg. As described before, the data were downloaded from the Radish Data



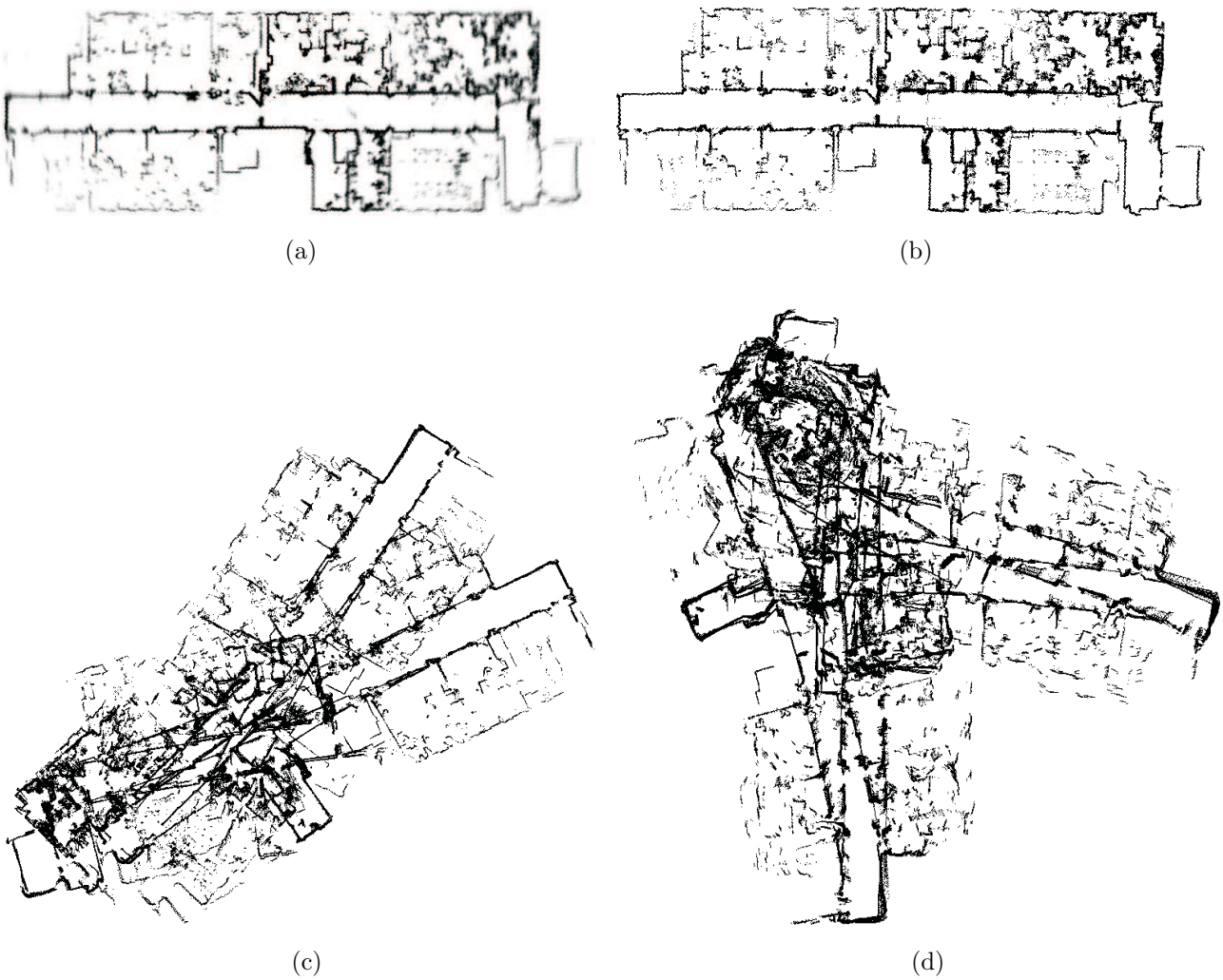
(a) (b)

(c) (d)

Figure 2.12.: Maps of the first floor of the iRP robotics laboratory with 3 cm resolution. a) RBSM average map computed over 20 iterations. b) Best result of RBSM c) Result provided by ICP d) Result provided by PSM

Set Repository. A Pioneer II robot equipped with a SICK laser gathered the sensor information. The laser was employed with a beam resolution of 0.5°. The AIS laboratory has a size of about 37 m × 14 m. The challenge of this data set is to keep the corridor straight in spite of the suboptimal odometry data. The map computed with the RBSM approach is depicted in Fig. 2.13b. It has a resolution of 5 cm. The result is very good although the top right part of the map exhibits noisy regions. This is probably due to people walking by and furniture building a cluttered environment. Since only an endpoint model was used instead of an occupancy grid, the maps become quite fast noisy because dynamic obstacles no longer disappear. The average map can be seen in

Fig. 2.13a and obviously the algorithm is also very stable in this scenario. In contrast, ICP and PSM completely fail to provide acceptable maps. The results computed by ICP and PSM are depicted in Fig. 2.13c and 2.13d. The maps have a resolution of 5 cm. It is obvious that both maps are useless for any further application. Similar to the intel data set, the rotational errors could not be compensated in a sufficient manner. Consequently, the same corridor appears several times in different directions.



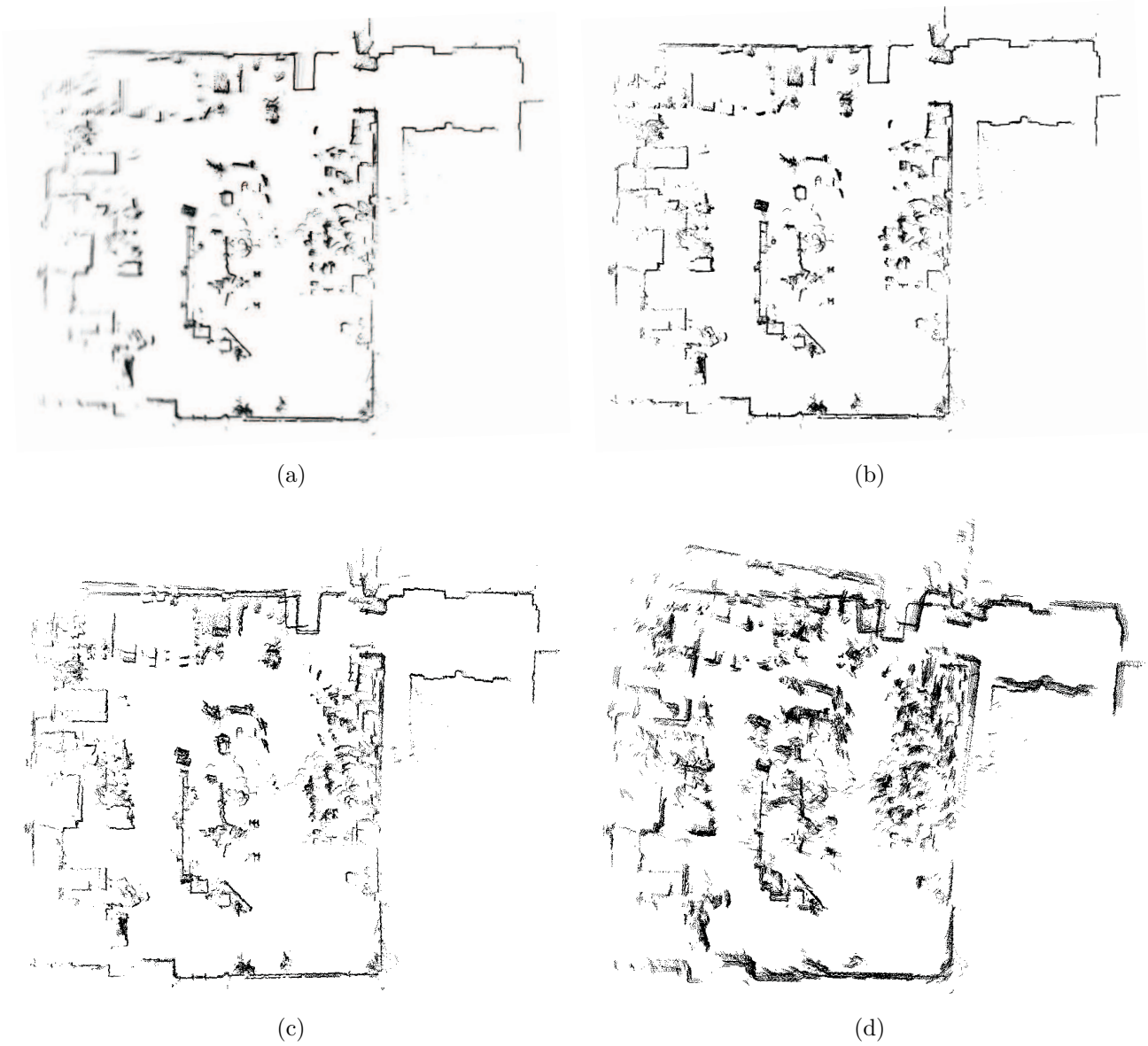
(a) (b)

(c) (d)

Figure 2.13.: Maps of the building 079 of the AIS laboratory of the University of Freiburg with 5 cm resolution. a) RBSM average map computed over 20 iterations. b) Best result of RBSM c) Result provided by ICP d) Result provided by PSM

The fourth experiment was conducted in the basement floor of the iRP robotics laboratory. The area has a dimension of 21 m $\times$ 18 m. The omnidirectional drive vehicle was employed again in order to record the sensor data. An observation was triggered as soon as the robot conducted a translational motion larger than 20 cm or after a maximal rotation of 10°. A typical map of this scenario computed by the RBSM method with 3 cm resolution is depicted in Fig. 2.14b. Since the laboratory contains many chairs, computer screens, and tables with industrial robots it represents a cluttered en-

vironment. Thus, bare walls and many different objects are registered in the map. The time required to collect the data was about 20 minutes whereas only 37 seconds were needed to compute the map. The average map is depicted in Fig. 2.14a. This data set is much easier to process than the previous ones because the loop which needs to be closed is relatively small and no walls subdivide the laboratory. Contrary to the second data set representing the first floor of the iRP laboratory, these conditions always allow for observing a sufficient portion of the laboratory with each scan. Hence, considerable inconsistencies are not very likely for the RBSM technique which is substantiated by the mean result. However, ICP and PSM fail to close the loop correctly (cf. Fig. 2.14c and 2.14d). It is clearly visible that the walls at the top of the maps appear twice in both cases. Again the rotational errors induced by the PSM matching routine are larger than the orientation discrepancies of ICP. Nevertheless, the results are of superior quality compared to data set 1 and 3. The maps have a resolution of 3 cm.



(a) (b)

(c) (d)

Figure 2.14.: Maps of the basement floor of the iRP robotics laboratory with 3 cm resolution. a) RBSM average map computed over 20 iterations. b) Best result of RBSM c) Result provided by ICP d) Result provided by PSM

2.5. Conclusion

In this chapter an efficient resampling based scan matching approach termed RBSM was presented. Basically, this method is highly comparable to a standard particle filter for tracking mobile robots, e.g. during path execution. In order to compute very accurate matching results, resampling takes place several times until a convergence criterion is satisfied. In each iteration, particles are drawn from a Gaussian distribution and re-weighted before the resampling operation is triggered. The (co)variance of the Gaussian is decreased after each iteration. Thus, the particles are densely spread around the most

likely true robot locations when the algorithm terminates. Consequently, the chance to obtain an accurate pose estimation is high. The convergence criterion exploits the effective sample size which is the inverse of the squared sum of the particle weights. The effective sample size indicates how reliable the particles are distributed in order to match the current observation. Intensive experiments compare the RBSM technique with a traditional trimmed ICP variant and with the PSM approach which is specially designed for laser scan matching. The experiments revealed that both ICP and PSM are of at least one order of magnitude faster than RBSM. In most cases, matching two consecutive observations yields comparable results while in rare cases ICP and PSM are even more precise. On the contrary, significant drifts from ground truth could be observed for both ICP and PSM in several scenarios. Not only the accuracy of matching two scans was investigated but the capability of building consistent maps of environments of considerable size was tested, too. The latter aspect is of higher importance in the context of this work. The largest test area had a size of 28 m $\times$ 28 m and RBSM was able to provide maps with accurately closed loops. However, ICP and PSM always failed completely since accumulated errors render a consistent map quite unlikely. In summary, ICP and PSM clearly outperformed RBSM concerning the matching time while RBSM is preferable when consistent maps have to be computed. ICP or PSM might be more useful in combination with a sophisticated SLAM algorithm when a rough and fast estimate of the current robot pose is more important than a reliable matching result.

References

[1] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 935–938, 1994.

[2] Mark Cummins and Paul Newman. Highly scalable appearance-only slam - fab-map 2.0. In *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[3] J. Gutmann. *Robuste Navigation autonomer mobiler Systeme*. Akademische Verlagsgesellschaft, 2000.

[4] I. J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. on Robotics and Automation*, 7(2):193–204, April 1991.

[5] K. Lingemann, H. Surmann, A. Nuechter, and J. Hertzberg. Indoor and outdoor localization for fast mobile robots. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2185 – 2190, September 2004.

[6] P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2743 – 2748, October 2003.

[7] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[8] J. Minguez, L. Montesano, and F. Lamiraux. Metric-based iterative closest point scan matching for sensor displacement estimation. *IEEE Trans. on Robotics*, 22(5):1047–1054, October 2006.

[9] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proc. of IEEE International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[10] L. Montesano, J. Minguez, and L. Montano. Probabilistic scan matching for motion estimation in unstructured environments. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3499 – 3504, August 2005.

[11] B. Jensen and R. Siegwart. Scan alignment with probabilistic distance metric. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2191 – 2196, September 2004.

[12] Albert Diosi and Lindsay Kleeman. Fast laser scan matching using polar coordinates. *The International Journal of Robotics Research*, 26(10):1125–1153, 2007.

[13] G. Weiss and E. V. Puttkamer. A mapbased on laserscans without geometric interpretation. In *Intelligent Autonomous Systems 4*, pages 403–407, 1995.

[14] A. Censi, L. Iocchi, and G. Grisetti. Scan matching in the hough domain. In *ICRA*, pages 2739 – 2744, April 2005.

[15] E. B. Olson. Real-time correlative scan matching. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 4387–4393, May 2009.

[16] D. Silver, D. Bradley, and S. Tayer. Scan matching for flooded subterranean voids. In *IEEE Conference on Robotics Automation and Mechatronics*, pages 422–427, 2004.

[17] R. Iser and F. Wahl. Building local metrical and global topological maps using efficient scan matching approaches. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1023–1030, 2008.

[18] Jun S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6:113–119, 1996.

[19] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. on Robotics*, 23(1):364–375, 2007.

[20] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 50(2):174–188, February 2002.

[21] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 2, pages 116 – 121, March 1985.

[22] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parametrization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics Science and Systems (RSS)*, 2007.

[23] A. Burguera, Y. Gonzalez, and G. Oliver. Probabilistic sonar scan matching for robust localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3154–3160, April 2007.

[24] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *Proc. of International Conference on Pattern Recognition*, volume 3, pages 545 – 548, 2002.

[25] K. M. Wurm, C. Stachniss, G. Grisetti, and W. Burgard. Improved simultaneous localization and mapping using a dual representation of the environment. In *Proc. of the 3rd European Conference on Mobile Robots*, pages 132–137, 2007.

Chapter 3

AntSLAM - Applying Swarm Intelligence to Global Map Optimization

3.1. Introduction

A very essential capability of mobile robots is to build maps of unknown environments which can be used in further applications, e.g. path planning and localization. In the literature, this problem is often referred to as the *Simultaneous Localization and Mapping* (SLAM) problem. It was addressed by many researchers in the past and lots of impressive results originated from their efforts. The SLAM problem is hard since it requires to estimate both the robot pose and the map at the same time yielding a high dimensional search space. In this thesis, a solution to the SLAM problem is required in order to construct the maps used to evaluate the localization algorithm presented in Chap. 5. Although consistent maps of small or even medium size workspaces were computed purely based on the scan matcher introduced in Chap. 2, a more sophisticated approach is necessary for generating large-scale maps, e.g. the map of the MIT Killian Court (cf. Fig. 3.11).

A first classifier for existing SLAM techniques may be the kind of representation of the environment. A popular method for map representation is to use fine grained grids, e.g. occupancy grid maps as introduced in Chap. 2 [1]. In contrast, a compact representation method comprises feature sets [2, 3, 4] which are very memory efficient. Topological maps consist of a set of nodes connected by edges on a logic level. Each node may represent an individual submap [5, 6, 7]. Finally, the appearance space was recently used by researchers to describe the environment [8, 9].

The second classifier criterion concerns the estimation paradigm, i.e. the philosophy of how to interpret the SLAM problem. An Extended Kalman Filter (EKF) was used in the pioneering work of Smith *et al.* [2]. More precisely, the robot pose and the location of all features were combined by a single mean and covariance. A drawback of this approach is that the update complexity is $O(n^2)$ where n is the number of features

maintained in the filter. EKF-SLAM computes the *online* posterior $p(\boldsymbol{y}_t \mid z_{1:t}, u_{1:t})$ [10]. Here $\boldsymbol{y}_t = \begin{pmatrix} \boldsymbol{x}_t \\ m \end{pmatrix}$ denotes the current SLAM state at time t. $\boldsymbol{x}_t$ represents the current robot pose and m is the active map hypothesis. $z_{1:t}$ and $u_{1:t}$ are measurements and control inputs.

Furthermore, particle filters were used in the past to cope with the SLAM problem. Murphy and Doucet *et al.* [11, 12] proposed *Rao-Blackwellized* particle filters (RBPF) as an effective mean to compute consistent maps. The underlying idea is to factorize the joint posterior $p(\boldsymbol{x}_{1:t}, m \mid z_{1:t}, u_{1:t})$ about the map m into $p(m \mid \boldsymbol{x}_{1:t}, z_{1:t}) \cdot p(\boldsymbol{x}_{1:t} \mid u_{1:t}, z_{1:t})$ [13]. The advantage is that only the trajectory of the robot needs to be estimated; but given the trajectory, the map can be computed very efficiently. Eliazar and Parr [14] introduced an efficient technique for map representation which allows to compute maps with several thousands of particles close to real time. On the contrary, Hähnel [15] proposed a RBPF employing an improved motion model with reduced variances. The idea was to derive a motion model based on estimating the uncertainty of scan matching operations. Howard [16] used the same motion model and expanded the idea of Rao-Blackwellization to multi-robot mapping. Grisetti *et al.* [13] devised an advanced proposal distribution which significantly reduced the number of required particles compared to the motion model of Hähnel [15]. All particle filter approaches mentioned so far have in common that they compute metrical maps. Montemerlo *et al.* [3] proposed FastSLAM which is a RBPF method but the map comprises features instead of discrete grid cells. The posterior is factored into one estimator for robot poses and an individual estimator for every feature of the map where each estimator constitutes an EKF. This approach was further optimized by the introduction of an enhanced proposal distribution taking the active measurement into account. Data association is handled by exploiting maximum likelihood estimation (MLE) techniques.

A very popular approach that *smoothes* a target function is named square root SAM (smoothing and mapping) proposed by Dellaert and Kaess [17]. This method encapsulates the SLAM problem in a belief network, i.e. it can be interpreted as a graph based technique smoothing the constraints of the edges. The objective is to recover the maximum a posteriori by minimizing the negative log-likelihood of the joint probability of the full SLAM state. Taylor expansion facilitates to transform the target function to a linear least-squares problem which is solved by applying a Cholesky factorization. In a subsequent publication, Kaess *et al.* [18] introduced iSAM which is a more efficient incremental version of square root SAM. New measurements are directly added to the squared information matrix instead of adding them to the full information matrix which avoids computational expensive factorization operations. Refer to Sec. 3.3 for further graph-based SLAM methods.

Recently, map building in the appearance space was intensively studied by researchers. Cummins and Newman [8, 19] introduced FAB-MAP which maintains a probability density over known places and unknown ones in the appearance space where the probability of observing a specific place is formulated as a recursive Bayes estimation problem. Kawewong *et al.* [9] used *position-invariant robust features* (PIRFs) to represented a

place. Their approach does not employ a probabilistic framework but relies on heuristic assumptions. The authors state that a higher recall-rate is achieved compared to FAB-MAP and dynamic objects are better filtered since PIRFs mainly capture objects that are likely to occur permanently in a scene.

Furthermore, *uncommon* ideas for map generation exist. For example, RatSLAM proposed by Milford and Gordon [20] simulates the spatial encoding in rodent brains and employs a simple webcam for environmental observation. They compute a *semi-metric* map, i.e. the map is self-consistent but does not necessarily reflect the real geometric properties of the environment. The results are impressive since a whole suburb was mapped using their approach. Howard *et al.* [21] represent the map as a manifold. This technique facilitates to represent the same location of the real world several times on the manifold, e.g. if the same loop is passed more than once, the map may implicitly look like a spiral stair. The map is subdivided into a set of *patches* and loop closing is postponed until enough information about the connectivity of the patches is available. The robot poses are projected onto a virtual horizontal plane and map optimization is performed by minimizing a least-squares error function derived from the network of robot poses.

Note that this list of references is far from exhaustive but provides a review of popular paradigms and techniques to handle the uncertainties which arise from the SLAM problem. In this study, an initial solution is computed using the scan matcher discussed in Chap. 2. This pre-solution is likely to exhibit critical errors but it is a proper basis for further optimization. The map is partitioned into a set of fragments which are connected by edges on a topological level. Furthermore, each edge contains a set of *samples* where each sample represents a possible transformation between consecutive fragments. Map correction is performed using a swarm intelligence algorithm which computes a sequence of samples keeping the map consistent. Thus, the proposed approach combines graph-based methods and biologically inspired techniques. In this approach, data association refers to the problem of correctly matching local maps when the robot revisits previously explored areas. Although many solutions were proposed in literature, the data association problem is partly ignored in the current implementation, i.e. map matching is either performed manually or using pRANSAM as explained in Chap. 4. The above mentioned optimization routine is realized employing the Ant Colony Optimization (ACO) meta-heuristic. Therefore, the proposed SLAM technique is referred to as AntSLAM. To the best of the author's knowledge, nobody applied ACO to the SLAM problem before. The objective of this chapter is to demonstrate an interesting, alternative optimization method to build globally consistent maps.

3.2. Related Works

As indicated in the previous section, the presented approach belongs to the class of graph-based SLAM methods. According to the tutorial of Grisetti *et al.* [22] graph-based SLAM approaches typically feature two components: techniques to derive the

graph from sensor measurements (the *front-end*) and a method to optimize the resulting constraint network (the *back-end*). Gutmann and Konolige [23] described a front-end for graph-based SLAM. Robot poses are topologically connected. The edge constraints are derived from scan matching operations. Loop closing is performed by correlating the current local map around the robot with older parts of the map. The size of the local patch depends on the uncertainty about the robot pose. Bosse *et al.* [5] introduced *Atlas*, a framework also constituting a front-end. In contrast to the work of Gutmann and Konolige [23] local maps are connected by edges instead of consecutive robot poses. The size of the local map is bounded by the number of features stored in the local map. Each local map carries a signature based on geometrical properties. A comparison operator for two signatures is defined which facilitates map matching utilized for loop closing. Finally the approach yields an error function but it is not stated how to minimize this error function. Estrada *et al.* [24] proposed both a front-end and back-end. An EKF is applied to compute a set of local maps which comprise features. A new map is initialized if the robot uncertainty with respect to the origin of the local map exceeds a fixed limit. Loops are kept consistent by employing sequential quadratic programming (SQP). A side condition is defined for the target function which assures loop consistency. *GraphSLAM* introduced by Thrun *et al.* [10] construct a target function derived from a belief network representation similar to the one of square root SAM [17]. However, they apply another optimization method which involves variable elimination, i.e. measurement nodes are removed from the network. Note that GraphSLAM is different from Graphical SLAM [25, 26] where the map is represented by a set of so called *energy nodes* resulting from odometry measurements and feature observations. The energy is minimized using a relaxation technique. Another popular work was published by Olson *et al.* [27]. A constraint network is generated, which is optimized using a stochastic gradient descent. Their work includes a dexterous node representation, which allows for efficient map update steps. Grisetti *et al.* [28] further improved the work of Olson by transforming the constraint network into a spanning tree which facilitates to update parts of the map individually without affecting the complete network. This approach fails if applied to the 3D case because linearization in $\mathcal{SO}(3)$ is necessary which is not a trivial manner since the underlying space is not Euclidean anymore. An extension to the 3D case was introduced by Grisetti *et al.* [29] where an optimization of the target function on a manifold is proposed taking the non-Euclidean structure into account.

The front-end approach presented in this chapter is most comparable to the work of Estrada *et al.* [24] and Bosse *et al.* [5] since both publications proposed to partition the global map into a set of independent local maps with topological connections between them. However, the back-end, i.e. the optimization routine is completely new.

3.3. Review of Ant Colony Optimization

For the sake of clarity, this section provides a brief review of ant colony optimization (ACO) which constitutes a very popular family of algorithms successfully applied to several NP-hard combinatorial optimization problems [30, 31] in the past. The basic

idea of this class of optimization techniques is inspired by the behavior of real ant colonies during foraging. While moving, ants are leaving pheromone trails permanently which are used for indirect communication amongst the ants. The pheromone is a mean for finding shortest paths between the nest of the colony and the forage. A short path contains more pheromone than a longer one because it can be passed more often in the same time. Starting from their nest, ants choose paths with a probability proportional to the strength of the pheromone. An example is given in Fig. 3.1 which illustrates the behavior of an ant colony. The upper left figure shows two ants starting from their nest at time $t = 0$. An obstacle blocks the way and two alternative routes exist to pass the obstacle. Let the upper route be twice as long as the lower one. The black ant chooses the longer path while the shorter one is chosen by the red ant. For the sake of simplicity, suppose the ants update their pheromone trail at discrete time steps. Since the lower path is twice as short as the upper one, the red ant reaches the forage at $t = 1$ and updates the pheromone to $\tau = 1$ (upper right). The lower left figure depicts the situation at $t = 2$. The black ant has reached the forage while the red ant moved back to the nest. Both ants accumulate the pheromone. Note that the shorter tour contains twice as much pheromone as the longer one. Consequently, more ants will decide to move the shorter tour in future time steps and thus the pheromone trails reflect the colony's experience [32]. As a result, at time $t = n$ (lower right) most ants select the bottom route and the tour at the top is rarely chosen. Dorigo *et al.* [33] proposed the Ant Colony Optimization meta-heuristic, which provides a framework for most ant algorithms. It is structured as follows [32]:

Algorithm 2 Ant Coloy Optimization

```
1: while termination condition not met do
2:     for i ← 1 to ANTS do
3:         Construct Solution
4:         Apply Local Search                          ▷ optional
5:     end for
6:     Update Pheromone Trails
7: end while
```

In the past, this technique has been applied to the famous Traveling Salesman Problem (TSP). Dorigo *et al.* [34] proposed Ant System that is the first ACO algorithm employed for solving the TSP. A city tour as in Algorithm (2) is constructed by putting each ant on a randomly chosen city at the beginning. Assuming ant k, $k \in [1; M]$, to be at city i, it chooses the next city j with probability [32]

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} , \tag{3.1}$$

where $\eta_{ij} = \frac{1}{d_{ij}}$ is a heuristic value. Here, d_{ij} is the distance between city i and city j. $\mathcal{N}_i^k$ is the set of cities which ant k has not yet visited and $\tau_{ij}(t)$ is the amount of pheromone of the edge connecting city i and j. The parameters α and β control the

relative influence of the pheromone strength and the heuristic value. In the case α=0 it is a simple greedy algorithm, which is very likely to choose the nearest city next. Conversely, β=0 leads to a fast stagnation, which means that all ants are following the same path resulting in a highly suboptimal solution in general [34]. Hence, a good trade-off between both variants needs to be found. After tour construction, the pheromone trails are updated by

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{M} \Delta\tau_{ij}^{k}(t) \tag{3.2}$$

where $0 < \rho \leq 1$ is an evaporation factor which allows for forgetting bad decisions made previously. $\Delta\tau_{ij}^{k}(t)$ is the new pheromone added by ant k and is defined as

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L^{k}(t)} & \text{if edge } (i,j) \text{ has been visited by ant } k \\ 0 & \text{otherwise ,} \end{cases} \tag{3.3}$$

where $L^{k}(t)$ is the length of the tour of ant k. Intensive research showed that Ant System is able to find good solutions only for relatively small TSP instances with 75 cities maximal. Ant System applied to larger instances lead to rather poor solutions. Thus, strong efforts were made to improve the algorithm. Dorigo and Gambardella [35] proposed Ant Colony System (ACS). In ACS, only the ant representing the best tour found so far is allowed to update the pheromone trail. Furthermore, with probability p' ant k moves to the city, for which $[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}$ is maximal when located at city i. Conversely, with probability $(1-p')$ it chooses the next city according to Eqn. (3.1). Ant Colony System has proven to give far better results for larger TSP instances than Ant System. Stützle *et al.* [36] introduced the $\mathcal{MAX} - \mathcal{MIN}$ Ant System. In this modification, both the iterative best and the global best ant are allowed to update the pheromone trail. Moreover, lower and upper limits for the pheromone strength are defined restricting the pheromone strength of all edges (i,j) to $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$. Experiments showed that the upper limit is very important since it avoids an unlimited increase of the pheromone strength on good tours and hence preventing the algorithm from search stagnation much better.

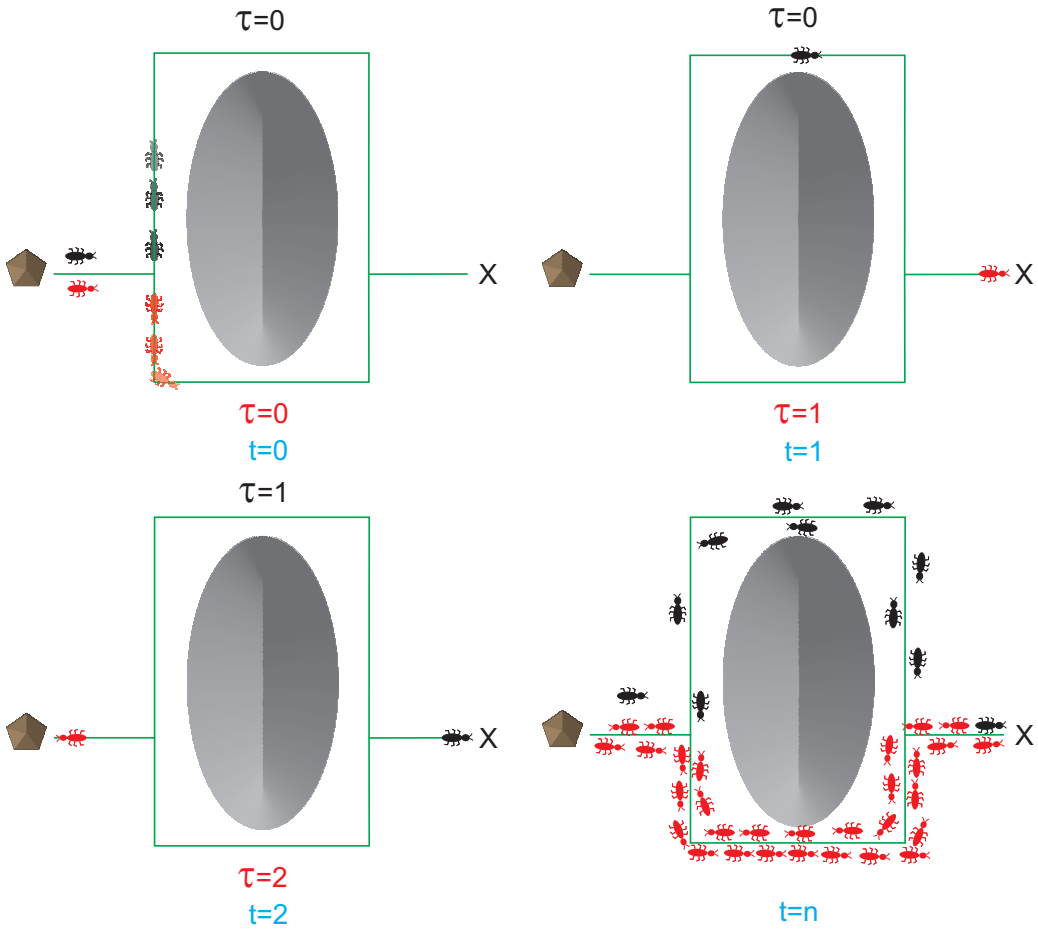


Figure 3.1.: Foraging of an ant colony. Each path used by the ants is marked with pheromone. Short paths are preferred since route alternatives are chosen with a probability proportional to the pheromone strength. Similar examples can be found in [34] and [35].

3.4. Application of Ant Colony System to SLAM

The ACS algorithm and its improvements described in the previous section are adapted in order to compute a consistent map on the global level. For this purpose, a graph $G = (V, E)$ with vertices V and edges E is necessary representing possible solutions to the SLAM problem. Furthermore, a weight is assigned to each edge of the graph reflecting its quality. Given an arbitrary sequence of nodes generated by the ants, the sum of the weights indicates a measure of confidence of the solution. The remainder of this section is organized as follows. The construction of the graph is explained first (cf. Sec. 3.4.1). Afterwards, Sec. 3.4.2 illustrates how to transfer ant colony optimization to the problem of global map correction.

3.4.1. Construction of the Graph

Suppose the robot starts the exploration of its unknown environment at the origin $(0, 0, 0)$ of a world frame W. It is equipped with a sensor providing range readings, e.g. a laser scanner. As already stated in the introduction, the global map is partitioned into a set of several fragments where each fragment represents a node in the aforementioned graph. Although pure scan matching causes inconsistencies in ever-growing maps, it is an efficient mean to solve the SLAM problem on a local level [5]. Hence, the scan matcher proposed in Chap. 2 is employed in order to compute the map fragments. More precisely, the local area around the robot is mapped until it has covered a predefined region. Note that other heuristics may be beneficially employed indicating the termination of the validity of the active map segment. For example Bosse *et al.* [5] utilize the robot pose uncertainty as an abortion criterion. However, the scan matcher applied in this study is very reliable. Thus, a simple distance threshold proved to be a proper heuristic in practice. When the exploration of the current map fragment terminates, a new one is initialized. As a result, a set of of map fragments is obtained, i.e. $\mathcal{F} = \{f_i \mid i \in [1, N], N \in \mathbb{N}\}$ where N is the total number of fragments computed so far. Thus, $V = \mathcal{F}$. The origin of f_i is represented by a local reference frame F_i. The subsequent fragment f_{i+1} is initialized with the reference frame F_{i+1} which is computed with respect to F_i, viz. the transition between two adjacent fragments is expressed by a homogenous transformation ${}^{F_i}\boldsymbol{T}_{F_{i+1}}$. Moreover, the following properties apply:

- The robot pose with respect to the new fragment is initialized as zero.
- Every subsequent motion entails a pose update with respect to all previous map frames, i.e. each fragment tracks the robot which may support future loop closing decisions. Additionally, the pose uncertainty is updated.
- The pose uncertainty with respect to the new fragment is set to zero [5].

Figure 3.2.: The global map is subdivided into a set of local maps or fragments. The origin of each fragment f_i is determined by a frame F_i. Adjacent fragments f_i and f_{i+1} are connected by transformations ${}^{F_i}\boldsymbol{T}_{F_{i+1}}$.

Tracking the robot pose with respect to a fragment f_i is performed by applying the standard update equations for the mean and the covariance [5, 10, 37]. Let $\boldsymbol{x}_{t-1}^i$ and $\boldsymbol{\Sigma}_{t-1}^i$ be the last robot pose and its covariance with respect to f_i. If a new motion command u_t arrives, the pose and the uncertainty are updated as follows:

$$\begin{aligned} \boldsymbol{x}_t^i &= g\left(u_t, \boldsymbol{x}_{t-1}^i\right) \\ \boldsymbol{\Sigma}_t^i &= \boldsymbol{J}_{x_{t-1}^i} \boldsymbol{\Sigma}_{t-1}^i \boldsymbol{J}_{x_{t-1}^i}^T + \boldsymbol{J}_{u_t} \boldsymbol{G}_t \boldsymbol{J}_{u_t}^T \end{aligned} \tag{3.4}$$

with

$$\boldsymbol{J}_{x_{t-1}^i} = \left.\frac{\partial \boldsymbol{x}_t^i}{\partial \boldsymbol{x}_{t-1}^i}\right|_{x_{t-1}^i, u_t}, \boldsymbol{J}_{u_t} = \left.\frac{\partial \boldsymbol{x}_t^i}{\partial u_t}\right|_{x_{t-1}^i, u_t}. \tag{3.5}$$

In Eqn. 3.4, the function $g(\cdot, \cdot)$ refers to a proper motion model while $\boldsymbol{G}_t$ is the covariance of the motion.

The principle of the map fragments is depicted in Fig. 3.2 which illustrates a robot mapping a corridor environment. The floor is subdivided into four fragments. The points of every second fragment are drawn red while the remaining map points are blue. Note that the fragments partially overlap. The frames highlighted in black represent the origins of the fragments. The current robot pose is depicted by the green circle in the left part of the map. Its orientation is marked by the green dash.

An important issue pertains the revisit of previously mapped fragments since in reality the robot might travel back to adjacent (known) places [5]. This implies that a constraint needs to be defined determining the active map fragment. To this end, so called *borders* are computed whenever the robot leaves the current place. Each border consists of two points, one to the left of the robot and one to the right. The line segment connecting both points determines the border. These points need to be selected carefully since the borders are the primary mean to decide in which fragment the robot is operating. The selection process includes the optimization of a target function based on the mixture of two criteria. Under the assumption that the robot moves along the principal axis of its local environment, the border should be nearly orthogonal to the vector defined by the orientation of the robot (cf. Fig. 3.3). The second criterion seeks to minimize the length of the vector in order avoid unnatural long borders which may be caused by erroneous sensor readings or small unmapped sections. The weight for each point $\boldsymbol{p}_j \in f_i$ is defined as

$$w\left(\boldsymbol{p}_j\right)_{1,2} = \alpha\eta_1 \mid\mid \boldsymbol{p}_j - \boldsymbol{p}_R \mid\mid + (1-\alpha)\eta_2 \mid \angle\left(\boldsymbol{p}_j - \boldsymbol{p}_R\right) - \Theta_r \pm \frac{\pi}{2} \mid \tag{3.6}$$

where $\eta_{1,2}$ are normalizers mapping the individual summands to $[0, 1]$. This is necessary to compare angles and Euclidean distances. The factor α determines the ratio of the

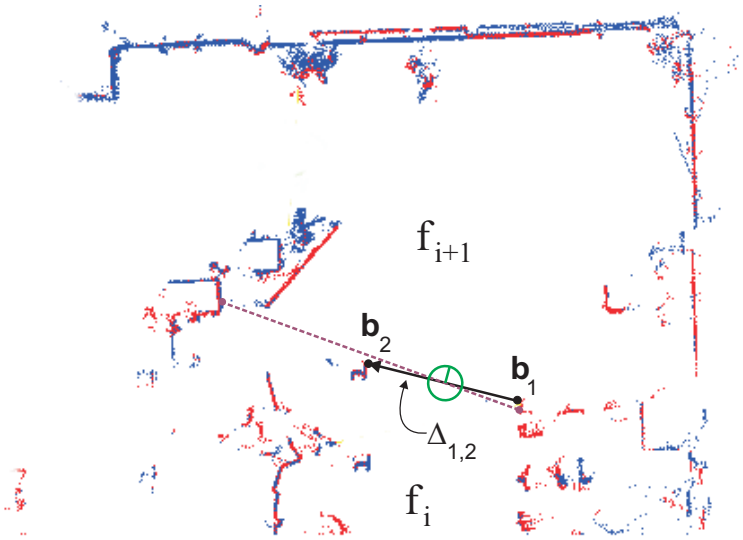


Figure 3.3.: The presence of the robot in a specific fragment is determined by means of *borders* consisting of two points $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ where is $\Delta_{1,2}$ is nearly orthogonal to $(cos(\Theta_r), sin(\Theta_r))$ where Θ_r is the robot orientation.

summands. $\angle$ denotes the angle of the vector $\boldsymbol{p}_j - \boldsymbol{p}_R$. Two weights are computed since each point $\boldsymbol{p}_j$ is rated as a potential candidate for both border points $\boldsymbol{b}_{1,2}$. The orthogonality criterion is considered by adding and subtracting $\frac{\pi}{2}$. The objective is to find the points minimizing $w\left(\boldsymbol{p}_j\right)_k$ for each $\boldsymbol{b}_k, k \in \{1, 2\}$:

$$\boldsymbol{b}_k = \underset{w(\boldsymbol{p}_j)_k}{\operatorname{argmin}} \{j \mid j \in [1, M_i]\} \tag{3.7}$$

where M_i is the number of map points of fragment f_i. Furthermore, each border has an orientation defined by the vector $\Delta_{1,2} = \overrightarrow{\boldsymbol{b}_1\boldsymbol{b}_2}$. An example is depicted in Fig. 3.3. The current robot pose is marked by the green circle with the dash. Considering only the orthogonality criterion, the border would be defined by the magenta colored dashed line segment. Obviously, the better choice are the points connected by the black line, i.e. the Euclidean distance influences the point selection yielding more rational results. The orientation is highlighted by the arrow.

If a new fragment is initialized, *two* borders are generated where the first one belongs to fragment f_i while the second one is assigned to f_{i+1}. Both borders are defined by the same points $\boldsymbol{b}_k$ but with inverse orientations and the points are expressed with respect to the individual frames. Thus, whenever the robot leaves f_i and enters unexplored terrain, a new border is assigned to f_i. Consequently, each fragment f_i holds a set of borders $\mathcal{B}_i = \{(\boldsymbol{b}_{j_1}, \boldsymbol{b}_{j_2}) \mid j \in 1...B_i\}$ where B_i is the total number of borders stored in f_i. The criterion determining whether the robot operates in fragment f_i is twofold. First, the position of the robot must be to the right of *all* Δ_{j_1,j_2} of f_i. Second, the robot position must be within the convex hull of f_i computed by using the Graham Scan algorithm [38]. Fig. 3.4 illustrates both criteria. The result of the first criterion is determined by

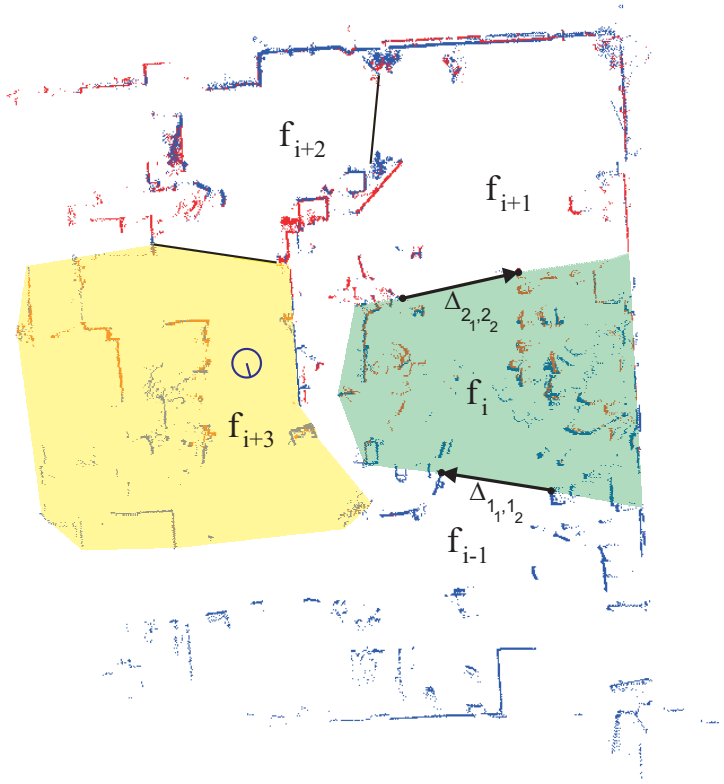


Figure 3.4.: The blue circle denotes the active robot pose. If only the function of Eqn. 3.8 was evaluated, the system would assume that the robot is located in fragment f_i but it is exploring a new fragment f_{i+3}. Therefore, the position of the robot must be within the convex hull of f_i as indicated by the green area. The yellow area indicates the region of fragment f_{i+3}.

evaluating the function

$$h\left(\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2\right) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$$
$$= \begin{cases} < 0 \text{ if } \boldsymbol{p_2} \text{ is right of } \overrightarrow{\boldsymbol{p}_0\boldsymbol{p}_1} \\ = 0 \text{ if the points are collinear} \\ > 0 \text{ if } \boldsymbol{p_2} \text{ is left of } \overrightarrow{\boldsymbol{p}_0\boldsymbol{p}_1}. \end{cases} \tag{3.8}$$

Here, $\boldsymbol{p}_2$ represents the robot pose while $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ constitute the border points. Inversely, the robot leaves the local map if the distance to *all* borders of the fragment is greater then the aforementioned threshold.

The active fragment may have two different states namely EXPLORE and TRAVERSAL. The state EXPLORE implies that mapping may be performed while the robot just passes the fragment if its state is TRAVERSAL. If the exploration of the current fragment terminates, the state is switched from EXPLORE to TRAVERSAL. A transition from TRAVERSAL to EXPLORE is forbidden in order to avoid an arbitrary growth of the local maps. Consequently, if a previously mapped place is revisited, the state

remains TRAVERSAL. Note that a fragment f_i does not necessarily have only a single adjacent node but a set of adjacent fragments.

Edge Computation

Basically, an edge e_{ij} between two adjacent fragment f_i and f_j is represented by the triple $e_{ij} = \left(f_i, f_j, {}^{F_i}\boldsymbol{T}_{F_j}\right)$ and thus the set of edges is defined as $\mathcal{E} = \{e_{ij} \mid i, j \in 1, ..., N\}$. Unfortunately, this is not sufficient since edge manipulation requires at least some measure of uncertainty. Moreover, the optimization approach proposed in this thesis employs an ant colony moving along the edges of the graph. Consequently, the transition from node i to j needs to be represented by several edges accounting for the uncertainty between the nodes. The computation of a consistent map on the global level requires slight modifications of the relative transformation of adjacent map fragments. Grisetti *et al.* [13] proposed a grid-based SLAM technique based on a highly improved proposal distribution [39] of a Rao-Blackwellized particle filter where each particle represents an individual robot trajectory. Given the trajectory, the map can be computed efficiently. To this end, the odometry information is corrected by employing a scan matching procedure. A closed-form approximation of the optimal proposal distribution is obtained by evaluating the sensor model for K sampled points around the corrected odometry estimation. Subsequently, Gaussian parameters are calculated using the sampled points and the next particle generation is drawn from the resulting normal distribution. The described idea was the main inspiration of this work to compute the edges of the graph; given the robot pose $\boldsymbol{x}_t^i$ at time t with respect to f_i, the odometry data are corrected applying a scan matching operation resulting in an improved pose estimation $\hat{\boldsymbol{x}}_t^i$. Then a set of samples $\mathcal{X} = \left\{\boldsymbol{x}_k \mid ||\boldsymbol{x}_k - \hat{\boldsymbol{x}}_t^i|| \leq \Delta, k \in [1, K]\right\}$ is generated around the scan matching result which allows for the computation of a Gaussian distribution $\mathcal{N}\left(\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij}\right)$ with the parameters

$$\boldsymbol{\mu}_{ij} = \eta \sum_{k=1}^{|\mathcal{X}|} \boldsymbol{x}_k \cdot p\left(z_t \mid f_{t-1}^i, \boldsymbol{x}_k\right) \tag{3.9}$$

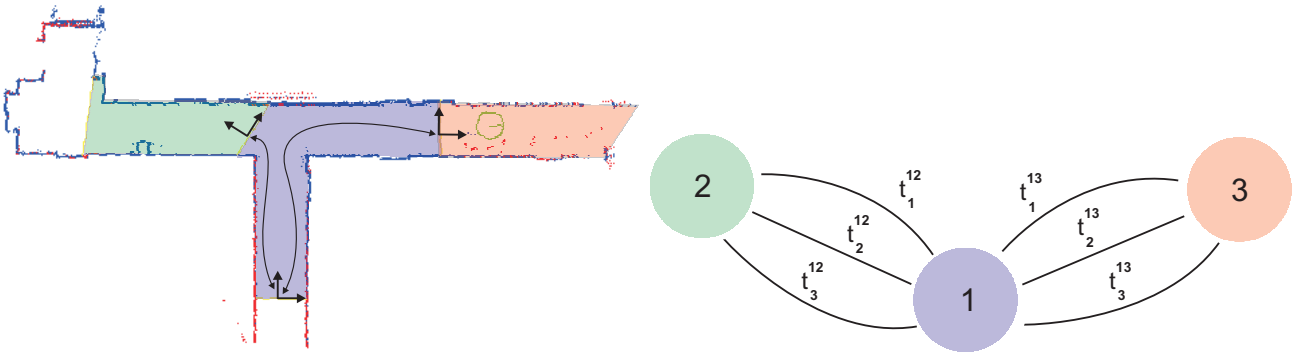
and

$$\boldsymbol{\Sigma}_{ij} = \eta \sum_{k=1}^{|\mathcal{X}|} (\boldsymbol{x}_k - \boldsymbol{\mu}_{ij})(\boldsymbol{x}_k - \boldsymbol{\mu}_{ij})^T \cdot p\left(z_t \mid f_{t-1}^i, \boldsymbol{x}_k\right). \tag{3.10}$$

Here, f_{t-1}^i is the map fragment at time $t-1$ before integrating the current sensor reading z_t and η constitutes a normalizer. Then, a second set of samples

$$\mathcal{T}_{ij} = \left\{\boldsymbol{t}_s^{ij} \mid \boldsymbol{t}_s^{ij} \sim \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij}), s \in [1, S]\right\} \tag{3.11}$$

is computed where each sample $\boldsymbol{t}_s^{ij}$ represents a potential transformation from fragment f_i to f_j. Note that $\boldsymbol{t}_s^{ij}$ is not a homogeneous matrix transformation but a vector. Thus,



(a) (b)

Figure 3.5.: a) A local map fragment may have several connected adjacent fragments. b) Edges are represented by a set of transformations. The inverse transformations leading back to node 1 are not depicted.

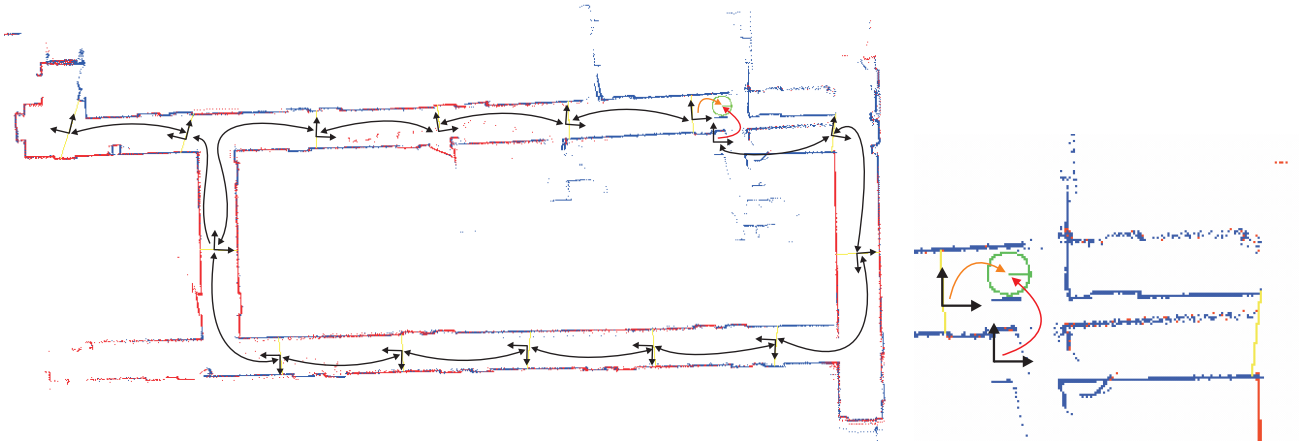
edges are redefined as $e_{ij} = (f_i, f_j, \mathcal{T}_{ij})$. Consequently, each edge e_{ij} provides several options to move from fragment f_i to f_j. Moreover, all edges are bidirectional, i.e. $e_{ij} \in \mathcal{E} \rightarrow e_{ji} \in \mathcal{E}$.

Fig. 3.5 depicts the topological structure of three adjacent fragments of the USC-SAL logfile[1]. A section of the map is shown in Fig. 3.5a. The edges are bidirectional. The edges connecting the fragments are represented by a set of transformations as outlined in Fig. 3.5b.

Loop Closing

A further issue subject to discussion pertains data association, i.e. place recognition which is mandatory for loop closing. Numerous techniques are proposed in the literature dealing with the problem of recognizing previously visited regions. Bosse *et al.* [5] compute a signature of each map fragment consisting of geometrical features and the signatures are compared in order to detect loop closing events. No pose estimation is employed in order to preselect loop closing candidates. Bosse and Zlot [40] introduced a novel keypoint descriptor in a metric space where weighted moments of oriented laser points around the local region of the keypoint are computed. A further well-known approach is the Joint Compatibility test [41] which uses the squared Mahalanobis distance to asses measurement features and map features. It takes the correlation of sensor measurements into account, i.e. a *greedy* strategy assigning the best map feature to each measurement feature is likely to cause spurious results. In this thesis, the robot pose with respect to the individual fragments is used in order to obtain a proper guess about potential loop closing areas worth of further investigation. The rational of exploiting the robot pose is that the scan matcher described in Chap. 2 yields results of proper quality. Let $\boldsymbol{p}^i_{min}$ be the point of fragment f_i which minimizes the Mahalanobis distance

[1] http://radish.sourceforge.net



(a) (b)

Figure 3.6.: a) Example of the topological structure of the map at the loop closing event. b) Orange arrow: robot pose with respect to the current map fragment; red arrow: robot pose with respect to the first fragment of the loop.

$$\sqrt{(\boldsymbol{x_t^i} - \boldsymbol{p}^i_{min})\boldsymbol{\Sigma}_t^{i\,-1}(\boldsymbol{x_t^i} - \boldsymbol{p}^i_{min})^T}$$

where $\boldsymbol{\Sigma}_t^i$ is the covariance of the robot pose. If the distance is below a fixed threshold the fragment is considered as a loop closing candidate. Subsequently, the active local map f_a is matched to f_i using pRANSAM which is described in Chap. 4. If the matching result is of acceptable quality, loop closing is performed, i.e. the hypotheses generated by pRANSAM is considered as the correct robot pose $\boldsymbol{h}_t^i$ with respect to f_i. However, this heuristic works well for the datasets investigated in previous publications [42, 43] but in general it is highly suboptimal since many places which are proper loop closing candidates may look similar yielding frequent erroneous map alignments. Thus, this part of the SLAM system is subject to further improvement, i.e. the current place recognition mechanism needs to be replaced by a more robust algorithm.

Given the corrected robot pose, the algorithm elucidated above for edge computation may be employed in order to connect the first and the last fragment of the current loop based on the Gaussian distribution $\mathcal{N}\left(\boldsymbol{\mu}_{h(a,i)}, \boldsymbol{\Sigma}_{h(a,i)}\right)$. As explained in Sec. 3.4.2, this Gaussian distribution is applied for the assessment of the map hypotheses computed in the optimization routine.

Fig. 3.6a depicts an example for a loop closing event. The topological structure is highlighted by the black arrows. The orange arrow indicates the robot pose with respect to the current map fragment while the red arrow represents the robot pose with respect to the first fragment of the loop. The close to optimal transformation $\boldsymbol{h}_t^i$ is known from the map matching operation. Thus, the quality of a map hypothesis may be assessed by evaluating the transformation marked by the red arrow. A detailed view is illustrated in Fig. 3.6b. The green circle indicates the robot pose.

In order to formalize the evaluation of a specific map hypothesis define

$$\mathcal{L} = \{f_{l_1}, ..., f_{l_L} \mid L \in \mathbb{N}\}$$

as the sequence of nodes constituting the loop where f_{l_1} is the first fragment while f_{l_L} is the last one. This sequence is determined by utilizing Dijkstra's shortest path algorithm [44]. In the current implementation the distance between two nodes is constant for all nodes of the graph, i.e. $\mathcal{L}$ is a set of minimal cardinality. Other strategies are possible, e.g. Bosse *et al.* [5] employ the uncertainty of the transformation of adjacent fragments as a distance measure. A map hypotheses is defined by a concrete choice of transformations $\mathcal{H} = \left\{ \boldsymbol{t}_{s(i)}^{l_i l_{i+1}} \mid i = 1, ..., L-1 \right\}$. Consequently, the robot pose with respect to the first fragment is computed as

$$^{F_{l_1}}\boldsymbol{T}_{x_t^{l_L}} = \left(\prod_{i=1}^{L-1} \boldsymbol{T}_{s(i)}^{l_i l_{i+1}} \right) . ^{F_{l_L}}\boldsymbol{T}_{x_t^{l_L}}. \tag{3.12}$$

where $\boldsymbol{T}_{s(i)}^{l_i l_{i+1}}$ is a homogeneous matrix transformation derived from $\boldsymbol{t}_{s(i)}^{l_i l_{i+1}}$. Refering to the example given in Fig. 3.6, $^{F_{l_1}}\boldsymbol{T}_{x_t^{l_L}}$ is represented by the red arrow while the orange one indicates $^{F_{l_L}}\boldsymbol{T}_{x_t^{l_L}}$. The hypothesis is evaluated employing

$$\begin{aligned} \mathcal{W}(\mathcal{H}) = \sum_{i=1}^{L-1} \left(\left(\boldsymbol{t}_{s(i)}^{l_i l_{i+1}} - \boldsymbol{\mu}_{i(i+1)} \right) \boldsymbol{\Sigma}_{i(i+1)}^{-1} \left(\boldsymbol{t}_{s(i)}^{l_i l_{i+1}} - \boldsymbol{\mu}_{i(i+1)} \right)^T \right) \\ + \left(\boldsymbol{x}_t^{l_1} - \boldsymbol{\mu}_{h(l_L,l_1)} \right) \boldsymbol{\Sigma}_{h(l_L,l_1)}^{-1} \left(\boldsymbol{x}_t^{l_1} - \boldsymbol{\mu}_{h(l_L,l_1)} \right)^T \rightarrow \min \end{aligned} \tag{3.13}$$

Note that $\boldsymbol{x}_t^{l_1}$ is the robot pose with respect to fragment f_{l_1} given the sequence of transformations stated above. A detailed mathematical derivation of the target function is given in appendix E. The objective is to find a map hypothesis $\mathcal{H}$ minimizing $\mathcal{W}(\mathcal{H})$.

3.4.2. Global Map Optimization

The optimization of the global map requires the introduction of a second graph where the nodes are represented by the sequence of fragments $\mathcal{L}$. The edges of consecutive fragments are given by $\mathcal{T}_{l_i,l_{i+1}}$, $i \in [1, L-1]$.

Fig. 3.7 depicts an example of the graph showing a tree-structure. Each level represents the set of transformations connecting two fragments. The edges connecting the bottom level and the last node denote the transformation from fragment f_{l_L} to the robot pose $\boldsymbol{x}_t^{l_1}$. As a result, each path from the root to the leaves composes a loop hypotheses, viz. the fragments involved in the sequence need to be manipulated such that loops can be closed in a consistent manner. It is important to note that the algorithm proposed in this chapter does not solve the full SLAM problem [10] but the map is corrected loop-wise.

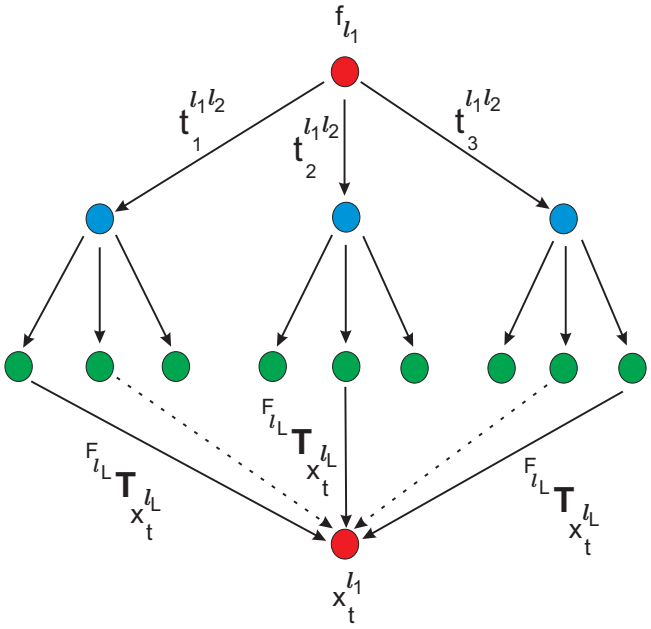


Figure 3.7.: The graph explored by the ant colony for map optimization comprises a tree structure. The transformation from the last node to the robot pose is the same for all hypotheses. Thus, a solution is determined by the path from the root to the leaves of the tree.

More precisely, once a fragment belongs to a corrected loop, it is not manipulated in later optimization procedures. This implies that the trees employed for map optimization may contain fixed transformations.

A ants are applied for seeking an optimal path through the tree. Let ant $k, k \in [1, A]$ be located at fragment f_{l_i}. Then it chooses the next edge $j \in \left[1, |\mathcal{T}_{l_i l_{i+1}}|\right]$ with probability

$$p_{l_{ij}}^k(t) = \eta^{-1} \cdot \tau_{l_{ij}}(t) \text{ where } \eta = \sum_{q=1}^{|\mathcal{T}_{l_i l_{i+1}}|} \tau_{l_{iq}}(t). \tag{3.14}$$

The amount of pheromone on edge j at time t is represented by $\tau_{l_{ij}}(t)$. The decision is not influenced by a heuristic information as in Eqn. 3.1. Each ant constructs its own hypotheses $\mathcal{H}^k(t)$ by selecting the edges step by step. According to Eqn. 3.2, the pheromone is updated by

$$\Delta\tau_{l_{ij}}^k(t) = \begin{cases} \mathcal{W}\left(\mathcal{H}^k(t)\right)^{-1} & \text{if edge } j \text{ of fragment } l_i \text{ has been} \\ & \text{chosen by ant } k \\ 0 & \text{otherwise .} \end{cases} \tag{3.15}$$

Thus, each ant may update the pheromone on the edges of the graph. However, this simple update strategy yields highly suboptimal results since the ants do not focus enough on the most promising solutions. Consequently, with a predefined frequency,

only the global best ant is allowed to update the pheromone [35]. In order to avoid search stagnation, with probability p_0 ant k chooses the next edge by drawing from an equal distribution. With probability $(1 - p_0)$ the selection rule 3.14 is applied. p_0 is initialized with $q_{0,min}$. It slightly increases in each iteration until $p_0 = p_{0,max}$. Thus, the higher the experience of the ant colony the more edges are chosen by employing the equal distribution. More important is the introduction of lower and upper bounds τ_{min} and τ_{max} which avoid an infinite pheromone growth and guarantee minimal selection probabilities of the edges. As a result, the proposed method is a combination of $\mathcal{MAX}-\mathcal{MIN}$ Ant System and Ant Colony System.

The optimization routine is summarized in Algorithm 3. The parameter set

$$\mathfrak{A} = \left\{\mathcal{T}_{l_i l_{i+1}}, \mathcal{N}\left(\boldsymbol{\mu}_{l_i l_{i+1}}, \boldsymbol{\Sigma}_{l_i l_{i+1}}\right), \mathcal{N}\left(\boldsymbol{\mu}_{h(l_L,l_1)}, \boldsymbol{\Sigma}_{h(l_L,l_1)}\right), i \in [1, L-1]\right\} \tag{3.16}$$

provides all necessary information processed by Algorithm 3. The variables are initialized first (line 2 to 7). The optimization procedure is executed until a maximum number of iterations is exceeded (line 8). Subsequently, each ant iterates over all fragments and generates its individual tour, i.e. it selects a set of edges resulting in a global map estimation. Afterwards, the quality of the current hypotheses is evaluated (line 22) and $\mathcal{W}_{best}$ and $\mathcal{H}_{best}$ are updated if needed (line 23 to 26). Line 29 to 33 update the pheromone on the edges, viz. either the global best ant may update the pheromone or the whole colony. At the end of each iteration p_0 is updated, too (line 34 to 36). Finally, the best edge combination is returned (line 39).

Algorithm 3 Ant SLAM

procedure OPTIMIZE($\mathfrak{A}$)
  $\mathcal{H}_{best} \leftarrow \emptyset$
  $\mathcal{W}_{best} \leftarrow \infty$
  for $i = 1$ to L **do**
    $\tau_{l_i j} \leftarrow \tau_{init}, j \in [1, S]$
  end for
  $iteration \leftarrow 0$
  while $iteration < iter_{max}$ **do**
    for $k = 1$ to A **do**
      $\mathcal{H}^k \leftarrow \emptyset$
      $\mathcal{W}\left(\mathcal{H}^k\right) \leftarrow \infty$
      for $i = 1$ to L **do**
        Choose random number p
        if $p > p_0$ **then**
          Choose next sample
          according to Eqn. (3.14)
        else
          Choose next sample
          with equal probability
        end if
      end for
      Compute $\mathcal{W}\left(\mathcal{H}^k\right)$ using Eqn. (3.13)
      if $\mathcal{W}_{best} < \mathcal{W}\left(\mathcal{H}^k\right)$ **then**
        $\mathcal{W}_{best} = \mathcal{W}\left(\mathcal{H}^k\right)$
        $\mathcal{H}_{best} = \mathcal{H}^k$
      end if
    end for
    $iteration \leftarrow iteration + 1$
    if $iteration$ mod $GlobBestFreq == 0$ **then**
      Let all ants update the pheromone trails
    else
      Employ only the global best ant
    end if
    $p_0 = \delta^{-1} p_0$ ▷ $\delta < 1$
    if $p_0 > p_{0,max}$ **then**
      $p_0 = q_{0,max}$
    end if
  end while
  return $\mathcal{H}_{best}$
end procedure

3.5. Experimental Results

The proposed algorithm was tested on a system equipped with an Intel Core 2 Duo E8300 'Wolfdale' processor running at $2.83GHz$, an ASUS P5Q-E motherboard, and 2 GB of

RAM with a clock rate of $1066MHz$. Moreover, a Windows XP OS with a Visual Studio 2005 compiler was employed. The data used to evaluate the method is available online from the Radish Data Set Repository [45]. Three different datasets were investigated. Due to many false data association results, map alignment in the context of loop closing was performed manually in this experiments. The optimization routine was configured as follows. A new fragment was initialized when the robot covered a distance of 5 m. General settings which were not changed during the experiments are depicted in Table 3.1. The parameter shown in line 3 implies that adjacent fragments were connected by

Parameter	**Value**
Ants	20
Iterations	5,000
$\|\mathcal{T}\|$	40
$p_{0,min}$	0.1
$p_{0,max}$	0.3
δ	$\frac{10}{11}$
τ_{min}	10^{-6}
τ_{max}	1

Table 3.1.: Fixed parameter of AntSLAM

$|\mathcal{T}| = 40$ edges (cf. Eqn. 3.11). Furthermore, four different configurations depicted in Table 3.2 were compared in the investigated scenarios. These configurations investigate

Config No.	$p_{0,min}$	**GlobBestFreq**
C1	0.1	3
C2	0.0	3
C3	0.1	100
C4	0.1	1

Table 3.2.: Configurations employed in the experiments

the mutual influence of the frequency of the equal distribution and the update frequency of the global best ant. The parameter $p_{0,min}$ is set to 0 in configuration C2, i.e. no ants draw the next edge from an equal distribution. In contrast, the equal distribution is enabled in C4 and the global best ant updates the pheromone most of the time. C4 completely disables the global best ant update. Subsequently, the characteristics of the datasets processed in this section are summarized. A SICK laser range-finder was used in each scenario in order to gather distance information. The statistics presented in this section emerge from averaging the result of 100 optimization trials per scenario.

- The *Intel Research Laboratory* has a size of 28 m × 28 m. The laser was configured with a beam resolution of 1°. Results are depicted in Fig. 3.8. The map hypothesis before closing the loop is shown in Fig. 3.8a while the complete map after processing and optimizing all data is illustrated in Fig. 3.8b. The map was partitioned into 12 fragments. It has a resolution of 4 cm. Given 40 transitions per edge yields 40^{11}

possible solutions. The evolution of the target function (Eqn. 3.13) plotted against the time and the variance for all configurations are depicted in Fig. 3.8c and 3.8d.

- The second scenario represents a corridor environment of the *USC-SAL Buidling* (University of Southern California). The area has a size of about 35 m $\times$ 11 m. The initial solution before loop closing is illustrated in Fig. 3.9a. A considerable map inconsistency can be observed in the upper right part of the map. It has a resolution of 4 cm and 12 fragments were generated again. As in the Intel dataset, a proper solution needs to be selected among 40^{11} options. The map after the optimization routine was triggered is depicted in Fig. 3.9b. The target function and the variance are highlighted in Figs. 3.9c and 3.9d.
- The data of the third scenario were collected in the *ACES Building* of the University of Texas in Austin. Results are depicted in Fig. 3.10. The floor dimensions are 45 m $\times$ 41 m. Fig. 3.10a shows the map after exploring the outer square. It exhibits a conspicuous inconsistency at the bottom since the corridor appears twice in the map. The error is corrected in Fig. 3.10b. The area was mapped with a resolution of 6 cm. The outer loop was partitioned into 14 fragments which results in 40^{13} map hypotheses. Due to the length of the loop a new fragment was initialized every 10 m. The characteristics of the optimization procedure are depicted in Figs. 3.9c and 3.9d. This environment is a good example where a loop-wise map optimization is not sufficient. If the robot moves to the middle part of the map where the corridors intersect, four loops are closed at the same time. Consequently, either a system estimating the full SLAM posterior is employed or the proposed back-end requires further enhancement.

All three scenarios exhibit nearly the same characteristics with respect to the evolution of the target function and the variance. Configuration C3 always yields the best results. This implies that the pheromone update of the global best ant has a powerful effect on the average quality of the solution. The second best result is obtained by C1. Both C1 and C3 draw edges from an equal distribution and employ the leading ant for pheromone update with a certain frequency. The usage of p_0 is disabled in configuration C2. A comparison of C1 and C2 reveals that the introduction of the equal distribution is a further mean to increase the average quality. Moreover, C2 exhibits the highest variance since search stagnation may occur sometimes which has a negative effect on the stability of the solution. C4 also yields suboptimal results since all ants contribute equally to the pheromone update and thus the ant colony does not focus on promising regions of the hypotheses space.

Fig. 3.11 depicts a map of the MIT Killian Court dataset. This place has a size of 250 m $\times$ 215 m. It is the largest area ever mapped employing AntSLAM as optimization routine. No statistics were evaluated for this dataset. The reason for showing this example is to demonstrate that AntSLAM is capable of mapping large scale environments.

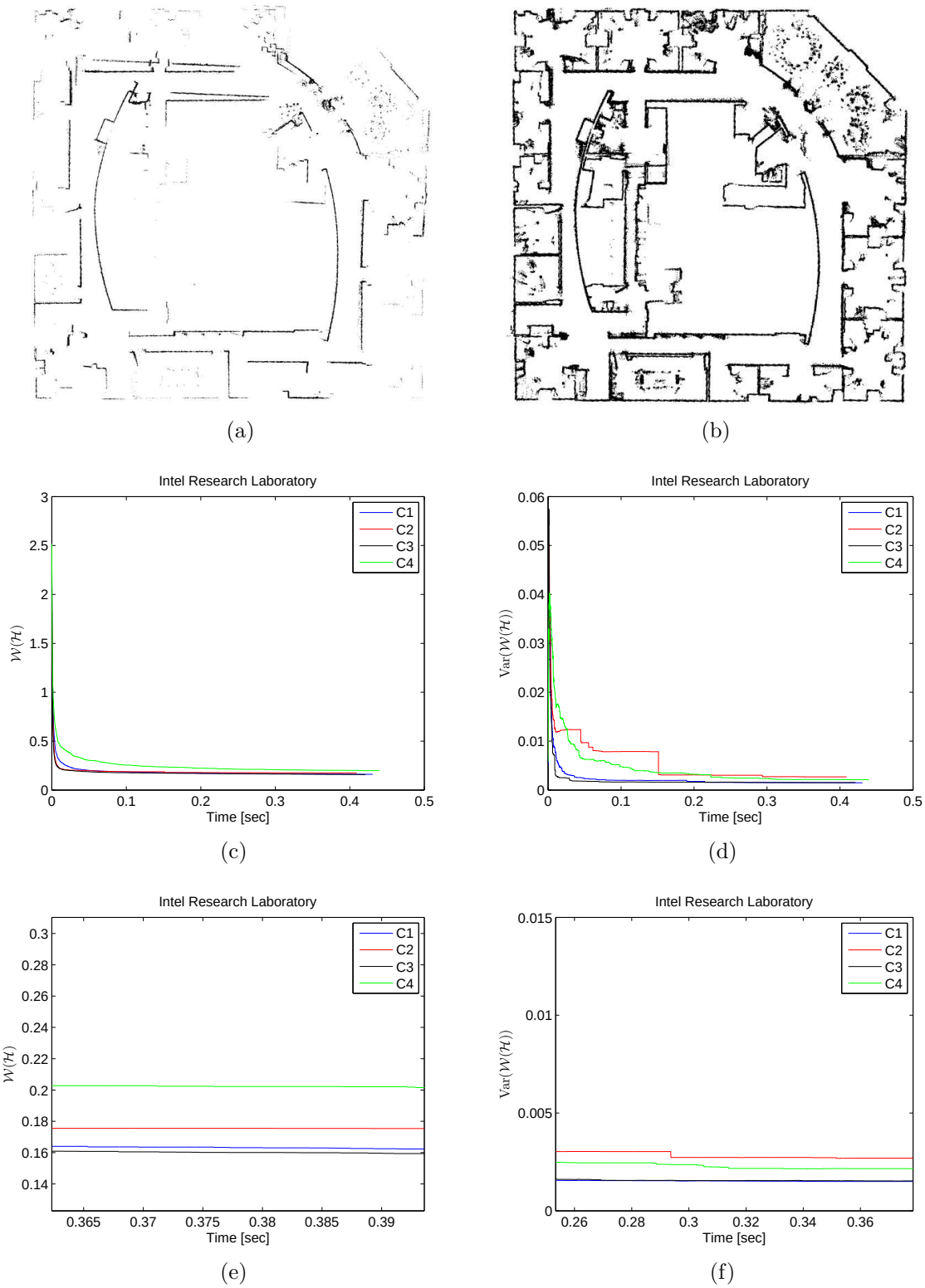


Figure 3.8.: Intel Research Laboratory. a) The map before closing the first loop (the upper corridor appears twice) b) The complete map c) The evolution of the map quality plotted against the time d) The variance of the map quality e) A detailed view of the map quality f) A detailed view of the variance

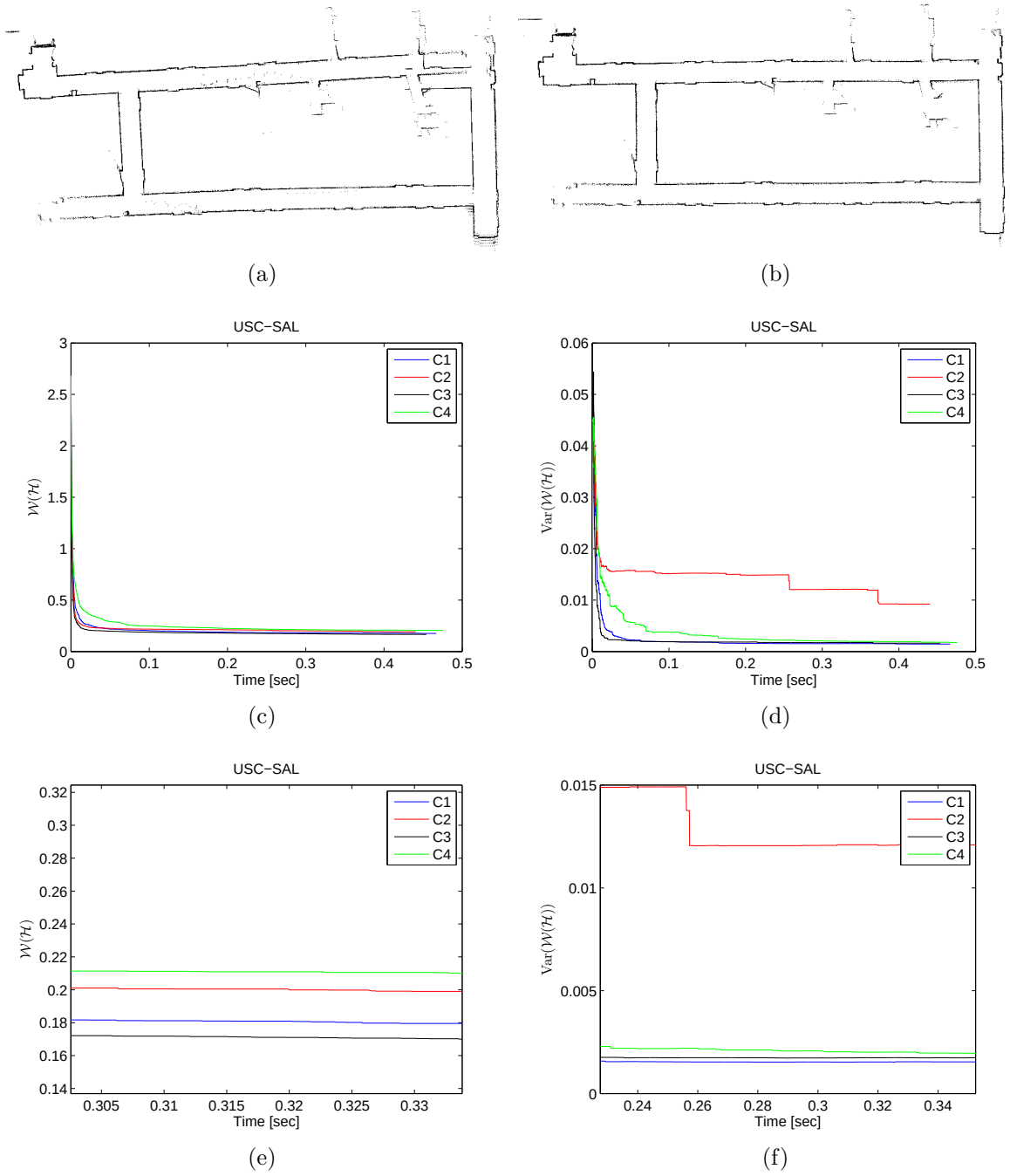


Figure 3.9.: USC-SAL Building. a) The map before closing the loop b) The map after correcting the inconsistencies c) The evolution of the map quality plotted against the time d) The variance of the map quality e) A detailed view of the map quality f) A detailed view of the variance

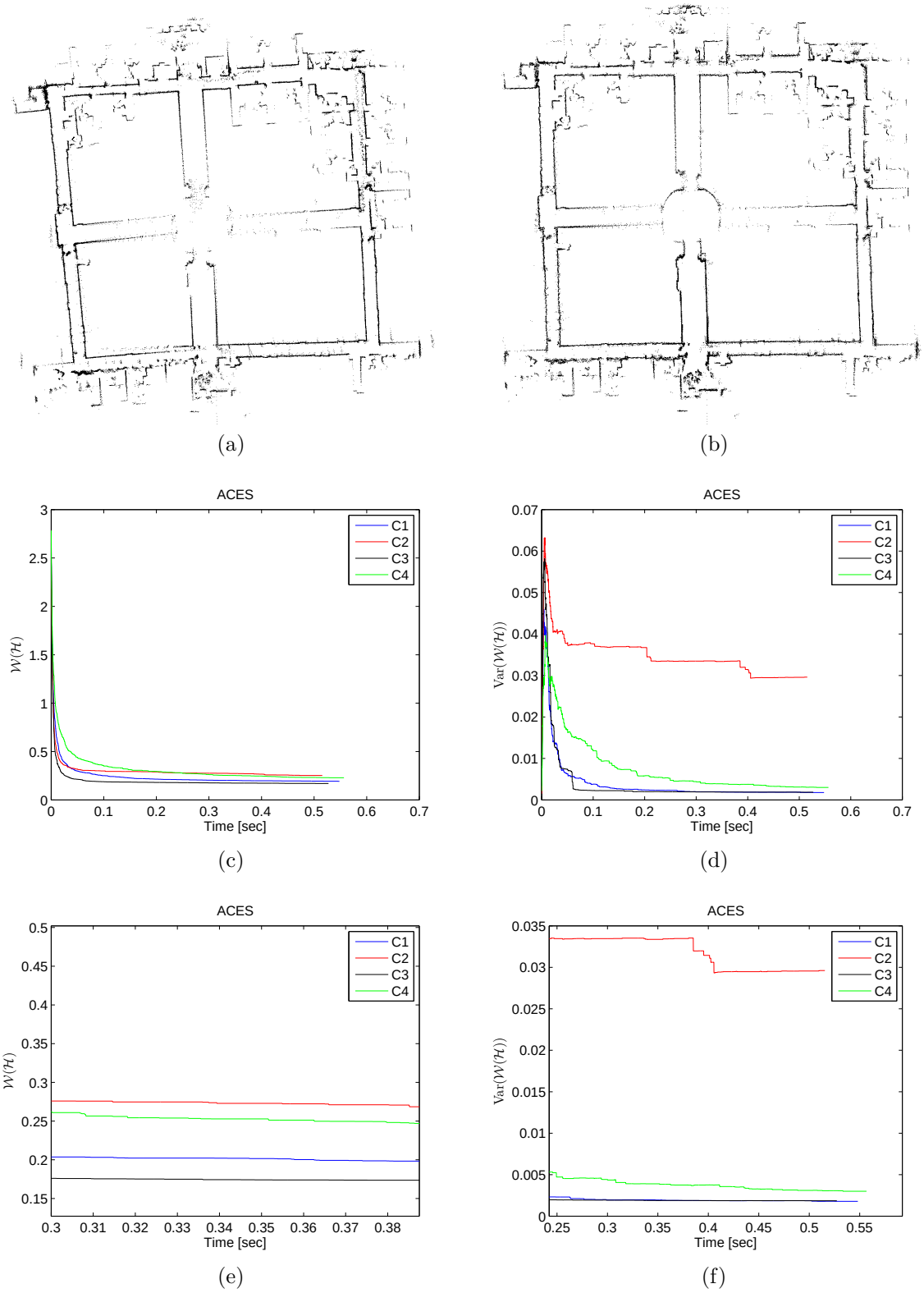


Figure 3.10.: ACES Building of the UT Austin. a) The map before closing the outer loop b) The map after correcting the inconsistencies c) The evolution of the map quality plotted against the time d) The variance of the map quality e) A detailed view of the map quality f) A detailed view of the variance

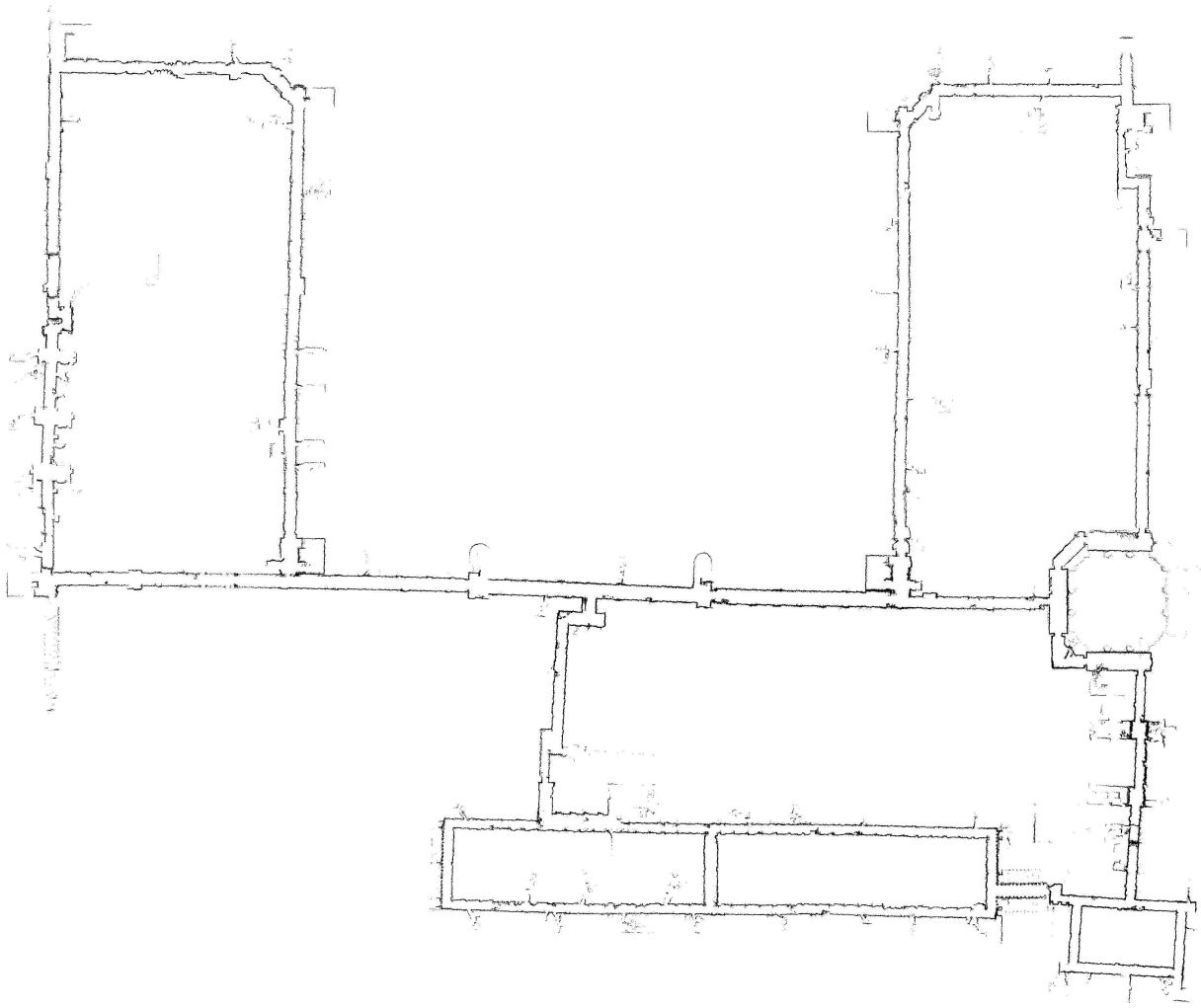


Figure 3.11.: The MIT Killian Court. The area has a size of about 250 m × 215 m.

3.5.1. Discussion

Many powerful techniques exist ranging from non-linear least square methods to gradient-descent based approaches which are able to minimize the target function 3.13. Therefore, it is interesting to compare the performance of AntSLAM with traditional optimization techniques. To this end, the target function 3.13 is minimized using *Levenberg-Marquardt* [46] which is a well known approach to solve non-linear least-square problems. Fig. 3.12 depicts several examples of the Levenberg-Marquardt[2] minimization. The characteristics of the optimization of the Intel map is shown in Fig. 3.12a. Obviously, LM converges conspicuously faster than the method proposed in this section (ACO), i.e. it terminates after $4 \cdot 10^{-2}$ seconds while ACO requires 0.4 seconds. A comparison of the curves implies that a fixed number of iterations is not a proper criterion to abort the computation of map hypotheses since ACO converges at nearly 0.1 seconds. Furthermore, the quality of the result is better. It is impossible for the proposed technique to provide results with

[2]abbreviated as *LM* in the remainder of this section.

arbitrary precision due to the discretized nature of the optimization paradigm. The correction of the USC-SAL map illustrated in Fig. 3.12b is the second example were LM outperforms the ACO optimization. Fig. 3.12c and 3.12d compose two examples where LM got stuck in local minima. Hence, ACO provides more accurate results in both cases. However, the suboptimal performance presented in these examples is a very rare event and depends on the initial solution. In summary, experiments showed that LM computes better solutions and thus more efficient least-squares methods may be preferable; but Fig. 3.12c and 3.12d prove that situations occur where ACO might be beneficially employed.

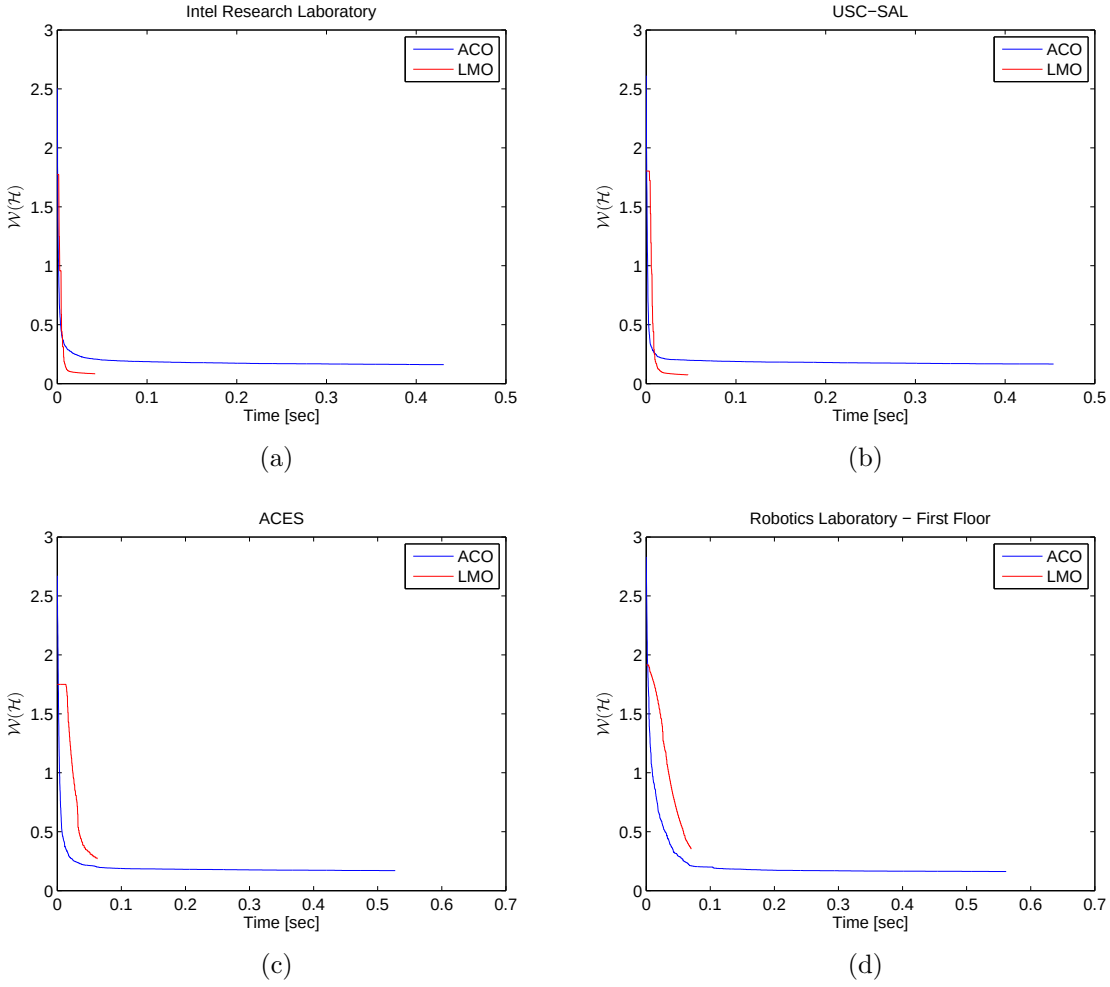


Figure 3.12.: Comparison of ACO and Levenberg-Marquardt optimization (LMO). Obviously, LMO outperforms ACO in terms of map quality in a) and b). c) and d) depicts two examples where LMO got stuck in local minima.

3.6. Conclusion

In this chapter, a method for solving the SLAM problem was presented which employs an optimization technique based on the Ant Colony Optimization meta-heuristic. In the past, ACO was successfully applied to the well known Traveling Salesman Problem. The scan matcher introduced in Chap. 2 was used to subdivide the map into a set of adjacent fragments connected by a set of edges on the topological level. Hence, the map is represented by metric means on the local level and as a graph with nodes and edges on the global level. In order to modify the relative transformation of consecutive fragments, each edge composes a set of samples drawn from Gaussian Distributions where each sample represents a map alignment hypothesis. Loop closing was performed by applying a map matching strategy (cf. Chap. 4) resulting in a hypothesis for the correct local map alignment. When a loop closing event was detected, Dijkstra's shortest path algorithm was employed for computing a sequence of nodes representing the shortest path between the first fragment of the loop and the last one. As a result, the ants explored merely a sub-graph of the whole topological structure when the optimization procedure was triggered. Although the solution provided by the ant colony may be arbitrarily far away from the optimal solution given a constant number of iterations, experimental results showed that consistent maps of the test environments could be computed with a proper accuracy. Unfortunately, the utilized map matching strategy for loop closing is often insufficient since frequent erroneous hypotheses were observed. Thus, this issue is subject to further improvement. Moreover, the SLAM system needs to be adapted to situations were several loops are closed at the same time. A comparison with a Levenberg-Marquardt optimization tool revealed that AntSLAM may be favorable in situations were least-square methods or gradient-descent based approaches get stuck in local minima. However, this turned out to be a rare event and thus other optimization methods may be preferable.

It would be interesting to apply the proposed method to other sensors like cameras or sonar which would require an adaption of the map representation as well as edge computation. For example, a mono SLAM system is attractive due to its marginal costs. Then the weight of an edge could be represented by the back projection error of depth information to the image plane.

References

[1] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, 1988.

[2] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, pages 167–193, 1990.

[3] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 2, pages 1985 – 1991, September 2003.

[4] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[5] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 2, pages 1899 – 1906, September 2003.

[6] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 5, pages 4845 – 4851, 2004.

[7] A. Angeli, S. Doncieux, S. J. A. Meyer, and D. Filliat. Incremental vision-based topological slam. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1031 –1036, September 2008.

[8] M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[9] A. Kawewong, N. Tongprasit, S. Tangruamsub, and O. Hasegawa. Online and incremental appearance-based slam in highly dynamic environments. *The International Journal of Robotics Research*, 2010.

[10] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. The MIT Press, 2005.

[11] K. Murphey. Baysian map learning in dynamic environments. In *Advances in Neural Information Processing Systems*, pages 1015–1021. MIT-Press, 1999.

[12] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 176–183. Morgan Kaufmann, 2000.

[13] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. on Robotics*, 23(1):364–375, 2007.

[14] A. Eliazar and R. Parr. *DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks*, volume 18, pages 1135 – 1142. 2003.

[15] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 206 – 211, October 2003.

[16] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[17] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[18] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics*, 24(6):1365–1378, December 2008.

[19] Mark Cummins and Paul Newman. Highly scalable appearance-only slam - fab-map 2.0. In *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

[20] M. Milford and G. Wyeth. Mapping a suburb with a single camera using a biologically inspired slam system. In *IEEE Trans. on Robotics*, volume 24, pages 1038–1053, 2008.

[21] A. Howard, G.S. Sukhatme, and M. J. Mataric. Multirobot simultaneous localization and mapping using manifold representations. *Proceedings of the IEEE*, 94(7), July 2006.

[22] G. Grisett, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31 –43, winter 2010.

[23] J. S. Gutmann, T. Weigel, and B. Nebel. Fast, accurate, and robust self-localization in polygonal environments. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1412 –1419, 1999.

[24] C. Estrada, J. Neira, and J. D. Tardos. Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Trans. on Robotics*, 21(4):588 – 596, August 2005.

[25] J. Folkesson, P. Jensfelt, and H. I. Christensen. Graphical slam using vision and

the measurement subspace. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 325 – 330, August 2005.

[26] J. Folkersson and H. I. Christensen. Closing the loop with graphical SLAM. *IEEE Trans. on Robotics*, 23(4):731–741, August 2007.

[27] S. Teller E. Olson, J. Leonard. Fast iterative optimization of pose graphs with poor inital estimates. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 2262–2269, 2006.

[28] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parametrization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics Science and Systems (RSS)*, 2007.

[29] G. Grisetti, C. Stachniss, and W. Burgard. Nonlinear constraint network optimization for efficient map learning. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3):428 –439, September 2009.

[30] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89(0):319–328, 1999.

[31] G. Di Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[32] T. Stuetzle and M. Dorigo. *Evolutionary algorithms in engineering and computer science: recent advances in genetic algorithms , evolution strategies, evolutionary programming, genetic programming and industrial applications*, chapter 9, pages 163–179. Wiley, 1999.

[33] T. Stuetzle and M. Dorigo. *The Ant Colony Optimization Meta-Heuristic*, chapter 9, pages 163–179. Wiley, 1999.

[34] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):1–13, 1996.

[35] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on on Evolutionary Computation*, 1(1):53–66, 1997.

[36] T. Stützle and H. H. Hoos. MAX-MIN ant system and local search for the travelling salesman problem. In *IEEE International Conference on Evolutionary Computation*, pages 309–314, 1997.

[37] A. Burguera, Y. Gonzalez, and G. Oliver. Probabilistic sonar scan matching for robust localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3154–3160, April 2007.

[38] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972.

[39] A. Doucet. On Sequential Simulation-Based Methods for Bayesian Filtering. Technical report, Cambridge University Department of Engineering, 1998.

[40] M. Bosse and R. Zlot. Keypoint design and evaluation for place recognition in 2d lidar maps. *Journal of Robotics and Autonomous Systems*, 57:1211–1224, December 2009.

[41] J. Neira and J. D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. on Robotics and Automation*, 17(6):890 –897, December 2001.

[42] R. Iser and F. Wahl. Building local metrical and global topological maps using efficient scan matching approaches. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1023–1030, 2008.

[43] R. Iser and F. Wahl. Antslam: Global map optimization using swarm intelligence. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 265–272, 2010.

[44] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[45] Radish: The robotics data set repository ⟨http://radish.sourceforge.net/⟩ [date: 11/04/2011]. Internet.

[46] J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin / Heidelberg, 1978.

Chapter 4

Global Point Cloud Matching

4.1. Introduction

In various robotics applications, matching several sets of points (also known as point clouds) is an essential component. One example is the field of service robotics. If a robot is supposed to move to a kitchen in order to pick up a cup of coffee, it has to know where the cup is, but of equal importance is to recognize the item. If both the kitchen and the cup are modeled as point clouds, object recognition can be realized by computing the best match of the model and the sensor data, e.g. by employing stereo vision data. Another application is to estimate the pose of objects based on tactile information. Industrial robots are typically equipped with a gripper to manipulate certain objects. Furthermore, the gripper may be configured with arrays of tactile elements. Application-related postprocessing of the tactile information could yield spatial point clouds. This information is often employed for object recognition [1] or estimating the current grip pose by matching the point cloud to a suitable model [2, 3].

Another important field where point cloud matching is used are medical robots. Nowadays robots often give assistance in surgery. For example, current research focuses on the repositioning of the bone fragments of broken femurs. The bone fragments are reconstructed by processing the data of a computer-tomogram. Matching the point cloud to a model facilitates to estimate the relative pose of the fragments [4]. The same methodology is applied to realign the fragments of broken pelvic bones [5].

Furthermore, point cloud matching is applied in the field of bioinformatics. Much research focuses on protein-docking since the interaction of proteins plays an important role in many biochemical processes. Point cloud matching is used to predict the protein interaction based on the geometrical complementarity of their surfaces [5].

In archeology one often has to cope with broken earthware or ceramics. The reconstruction of the shivered finds may be impossible for humans due to the nearly boundless number of possibilities to combine the fragments. Powerful matching algorithms assist the archaeologists to puzzle the shivers together [5].

In mobile robotics global point cloud matching is exploited to solve the global localization

problem. In Chap. 2 a rough estimate of the current robot pose was given but in the context of global localization no such estimate is available. Given a map of the workspace of the robot, localization is achieved by matching the sensor data to the map.

Formally, the matching problem is often defined as follows: Given two arbitrary point sets A and B with respect to a world frame $\mathcal{W}$, find a translation $\boldsymbol{t}$ and a rotation $\boldsymbol{R}$ resulting in a maximum overlap of these two point sets. The formal problem definition is closely related to the scan matching problem, but the initial pose estimation is missing.

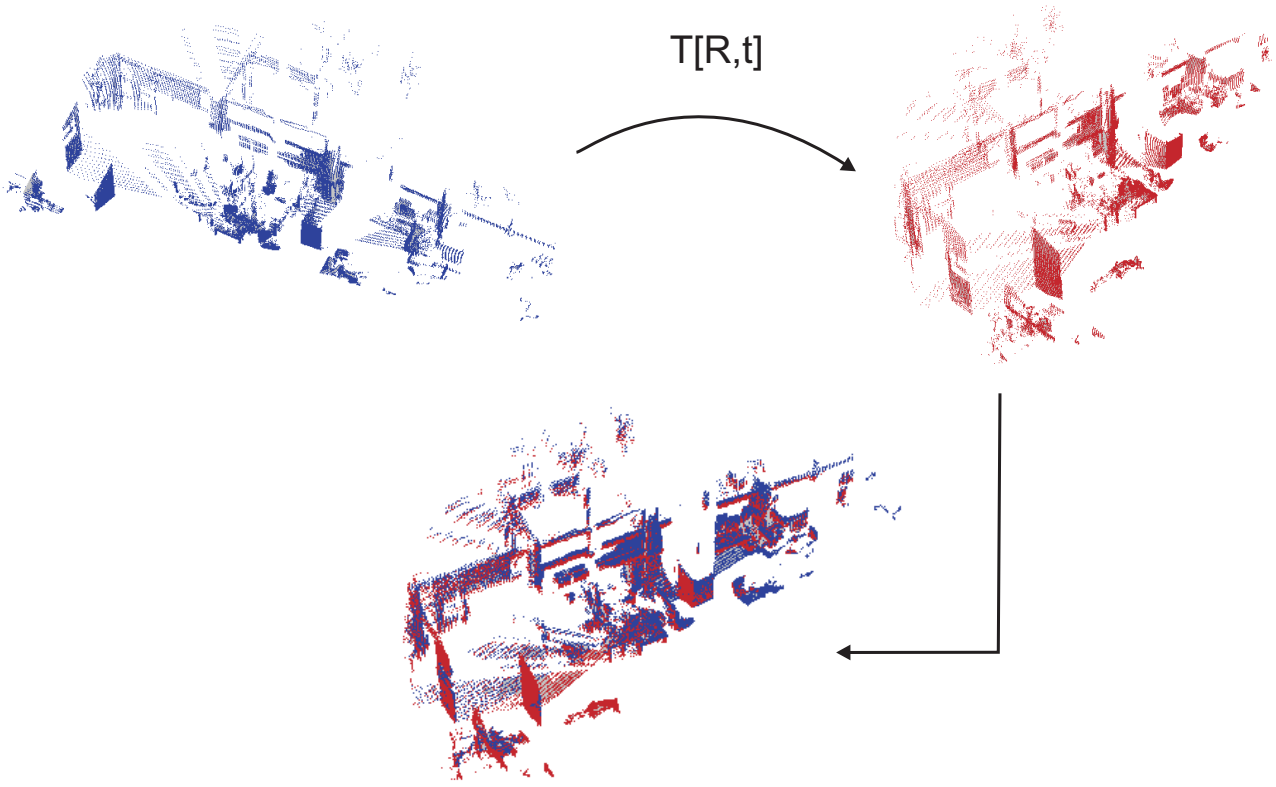


Figure 4.1.: Two perspectives of the iRP laboratory. The transformation T consisting of a rotation R and a translation t accurately overlaps both point clouds.

An example is depicted in Fig. 4.1. A 3D representation of the iRP robotics laboratory (cf. Chap. 2) from two different perspectives can be seen. The transformation T perfectly matches both point clouds. The remainder of this chapter is organized as follows. In Sec. 4.2 previous research is discussed. In Sec. 4.3 the Random Sample Matching (RANSAM) approach is briefly reviewed which is a very powerful matching algorithm [6]. An efficient parallelization of RANSAM - called pRANSAM - is introduced in Sec. 4.4 and its characteristics are described and formally proven. It is elucidated how calculations can be distributed among the available processors. In Sec. 4.5 experiments demonstrating the efficiency of the proposed approach as well as validating the theoretically derived characteristics are presented. Finally, Sec. 4.6 concludes the chapter. The work presented in this chapter was published previously in close cooperation with Daniel Kubus [7].

4.2. Related Works

One of the most popular matching algorithms is the Iterative Closest Point (ICP) algorithm (cf. Chap. 2) [8]. This approach minimizes the squared distance between points of two point clouds by iteratively computing a homogeneous transformation $(\boldsymbol{R}, \boldsymbol{t})$ between the point sets until convergence. It was proven that ICP converges at least to a local minimum. The computation of the transformation matrix requires to identify point-to-point correspondences. The generation of these point pairs consumes significant computational resources. Langis *et al.* [9] proposed a parallel version of the ICP algorithm which they called pICP. They presented a sophisticated algebraic transformation of the cross-covariance matrix of the point pairs to an expression which facilitates the propagation of the point sets from a parent process to a set of N parallel child processes. Then each process computes a fraction of the matrix and propagates its individual result back to the parent process. Finally, the parent process generates the global cross-covariance matrix that is essential for the rigid-body transformation. Langis *et al.* [9] observed a compelling speedup compared to the original ICP technique.

Nüchter [10] utilized pICP in his approach to the *Simultaneous Localization and Mapping* (SLAM) problem in 6D. The implementation of pICP was optimized by using the OpenMP framework. A SICK laser scanner was applied to generate 3D scans, which were matched incrementally by pICP. A parallelized version of the GraphSLAM method (LUM) [11] – called pLUM – was employed subsequently for global relaxation. A maximum speedup of 1.48 and an average speedup of 1.25 could be observed on a dual core machine.

Another very popular approach to matching problems is the Random Sample Consensus (RANSAC) method proposed by Fischler *et al.* [12]. It randomly generates a set of hypotheses by matching a minimal portion of both point sets and scoring each hypothesis by counting the number of matchable points. Several enhancements of the original RANSAC algorithm can be found in the literature. Nister [13] proposed a preemptive RANSAC scheme where a fixed number of hypotheses is generated. An efficient decision rule is introduced to select the best hypothesis which is chosen from a fixed-size hypotheses set. The main focus of this enhancement are real time vision applications. However, if the fraction of inliers is too small, this algorithm will fail to provide a good solution. Chum and Matas [14] proposed the Randomized RANSAC (R-Ransac) algorithm. The main idea is to randomize the hypothesis evaluation step, viz. a small subset of the data points is randomly drawn and a statistical test is performed. If all points of the subset pass this test, the complete data set is evaluated. R-RANSAC was further improved by Chum and Matas [15] by deriving a sophisticated termination criterion. Hypotheses generation and evaluation aborts as soon as the probability of finding a better solution than the current best solution generated so far drops below a pre-defined threshold. Thus, a given confidence level is exceeded. Moreover, the bail-out test devised by Chapel [16] terminates the evaluation step as soon as the probability that the current hypothesis gets a better score than the current best hypothesis falls below a fixed threshold. The idea is comparable to the extrapolation employed by RANSAM (cf. Sec. 4.3).

The applications of RANSAC are manifold. In particular, it was used in the field of mobile localization. Tanaka and Kondo [17] localized a mobile robot by matching a set of observations to a global map using a modification of the preemptive RANSAC scheme mentioned above. Lowe *et al.* [18, 19] also apply RANSAC for localization by matching features extracted from a camera image to a set of features stored in a data base. For more references to RANSAC-based mobile robot localization refer to Chap. 5.

In this chapter a parallelized variant of the Random Sample Matching (RANSAM) approach introduced by Winkelbach *et al.* [6] is described which is an efficient enhancement of the above mentioned RANSAC technique [12]. RANSAM exploits the theory of the so-called birthday attack which is known from cryptography [20] and determines an upper bound for the probability of a collision in a hash table. This technique was originally developed for solving the 3D-puzzle problem [6], e.g. matching a set of fragments of a broken object. The approach was successfully applied by Westphal [4] in medical robotics to estimate the relative pose of two bone fragments of a broken femur. Koenig [21] used RANSAM to estimate the gripping pose of an object by reconstructing the object surface based on tactile information. Then, the reconstructed surface is matched to a data base. Iser *et al.* [22] localized a mobile robot by matching the current observation to a map of the environment. Moreover, Iser *et al.* [23] proposed a solution to the SLAM problem where a set of local maps is connected to a topological map. The RANSAM technique is part of the loop-closing detection system. In this thesis, the parallelized variant of RANSAM is employed for global localization of mobile robots by matching an incrementally computed local map to a global map. A detailed description of the algorithm is provided in Chap. 5.

4.3. Review of the Random Sample Matching (RANSAM) - Approach

In the following section a brief summary of the algorithm is given. For further information please consult [5, 6].

Let $A = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$ and $B = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$ be two arbitrary point sets (represented by kd-trees) given with respect to a world frame $\mathcal{W}$. The matching operation works as follows. First, three points $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \in A, i, j, k \in [1, m]$ are sampled from a uniform distribution as long as the points are not located on a line. Consequently, these points form a triangle and define a 3D relation $rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ determined by their l^2-norm, i.e.

$$rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) = \begin{bmatrix} ||\boldsymbol{x}_i - \boldsymbol{x}_j|| \\ ||\boldsymbol{x}_j - \boldsymbol{x}_k|| \\ ||\boldsymbol{x}_k - \boldsymbol{x}_i|| \end{bmatrix}. \tag{4.1}$$

In the original work of Winkelbach [6] a more sophisticated four dimensional relation is applied, which differs completely from the relation defined in Eqn. 4.1. Winkelbach defined the relation by exploiting surface normals and their geometrical properties. For

efficiency reasons no surface normals are used in this thesis for robot localization. Instead, Eqn. 4.1 facilitates to store the points in a three dimensional hash table denoted by R_A:

$$R_A[rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)] = (\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) \tag{4.2}$$

Subsequently, the same process is repeated for set B: Three points $\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r \in B, p, q, r \in [1, n]$ are drawn randomly and stored in a second hash table R_B:

$$R_B[rel(\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r)] = (\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r). \tag{4.3}$$

This process is repeated alternatingly until a collision occurs, i.e.

$$rel(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) = rel(\boldsymbol{y}_p, \boldsymbol{y}_q, \boldsymbol{y}_r) \tag{4.4}$$

In this case a pose hypothesis representing the relative transformation between A and B is obtained. The run-time complexity to compute the first hypothesis is $O(n)$ where n is the size of the hash table. Assuming rel to be unique given a concrete point triple, only $1.2 \cdot \sqrt{n}$ triples need to be processed until a collision occurs with a probability of $p > 0.5$. This value is derived from the theory of the birthday attack which is known from cryptography [20]. The process converges to $O(1)$ because the hash table is continuously filled with new relations. A pose hypothesis $\boldsymbol{H}$ can be obtained by computing the centroid of each triangle defining two homogeneous transformation matrices where the position vectors correspond to the centroids.

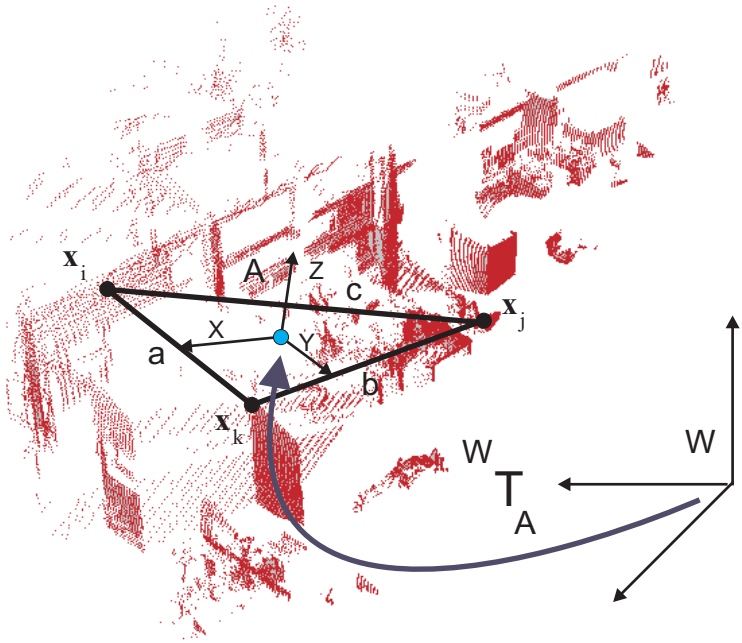


Figure 4.2.: Computation of ${}^{\mathcal{W}}\boldsymbol{T}_A$. The x-axis is determined by the vector connecting the centroid of the triangle and the middle of the triangle side of minimal length. The z-axis corresponds to the normal vector perpendicular to the plane defined by $\boldsymbol{x}_i$,$\boldsymbol{x}_j$, and $\boldsymbol{x}_k$. The y-axis follows from x and z.

The transformation from $\mathcal{W}$ to the centroid of the triangle of A is denoted by ${}^{\mathcal{W}}\boldsymbol{T}_A$; the transformation to the triangle frame of B is denoted by ${}^{\mathcal{W}}\boldsymbol{T}_B$. The principle of the

calculation of ${}^{\mathcal{W}}\boldsymbol{T}_A$ is depicted in Fig. 4.2. Now each point $\boldsymbol{x}_l \in A$ is transformed by the following equation:

$$\boldsymbol{x}_l{}' = {}^{\mathcal{W}}\boldsymbol{T}_B \cdot {}^{\mathcal{W}}\boldsymbol{T}_A{}^{-1} \cdot \boldsymbol{x}_l \tag{4.5}$$

Subsequently, the quality of the hypothesis is checked. This implies that a matching quality Ω needs to be defined as a function

$$\Omega = f\left({}^{A}\boldsymbol{H}_B, A, B\right) \rightarrow \mathbb{R} \in [0,1]. \tag{4.6}$$

In principle, this can be done by checking for each point of A whether the distance to its closest point in B drops below a threshold ε (cf. Eqn. 4.31). Hence, an appropriate measure of quality would be

$$\Omega = \frac{\sum_{i=1}^{|A|} contact_B(\boldsymbol{x}_i)}{|A|}. \tag{4.7}$$

If Ω either exceeds a predefined threshold Ω_{thres} or a maximum number of iterations is reached, the computation of hypotheses is aborted. The calculation of Ω can be speeded up considerably by using an extrapolation [6]. Instead of considering each point of A in every hypothesis check, a confidence interval within which the actual value of Ω lies with a probability of 0.95, i.e.

$$\Omega \approx \Omega_{\pm}(k) = \frac{\sum_{i=1}^{k} contact_B(\boldsymbol{x}_i)}{k} \pm \frac{1.96}{2 \cdot \sqrt{k}} \tag{4.8}$$

may be beneficially employed. k is the number of randomly drawn points of set A that are considered in a single extrapolation step. If the upper bound $\Omega_+(k)$ of the interval is lower than the quality of the current best match Ω_{best}, the hypothesis is discarded. Otherwise further extrapolation steps are performed until either $\Omega_+(k) < \Omega_{best}$ or all points of A have been considered. Note that good hypotheses may be discarded due to the randomized structure of the extrapolation technique, but this behavior does not influence the average quality of the matching result since a large number of hypotheses is checked.

4.4. pRANSAM - An Efficient Parallel Approach to Random Sample Matching

In contrast to the original RANSAM algorithm, pRANSAM exploits the advantages of systems with multiple processing nodes. The parallelization raises several questions. How can the parallelization be realized from a structural point of view? This issue is discussed in Sec. 4.4.1 and the approach to a parallel RANSAM algorithm is presented. Another interesting issue is whether there exists an optimal distribution of the algorithmic tasks to the processing node. Hence, Sec. 4.4.2 addresses important characteristics

of pRANSAM. Furthermore, real-time issues are briefly discussed (cf. Sec. 4.4.3). This is a relevant aspect for the practicability of pRANSAM in the mobile robot localization framework introduced in Chap. 5. If the matching routine is too slow its application is not feasible since localization is typically performed online.

4.4.1. Options of Parallelization

The RANSAM method is predestinated for parallelization. Nevertheless, concerning the implementation of a parallel RANSAM approach for a system with C processing nodes, several structural alternatives exist. The individual steps (point sampling, collision detection, and hypothesis evaluation) of the algorithm may be distributed to $T \geq C$ threads where one thread or a group of threads executes a particular step. For example, consider $T = 2$ threads. Then the first thread Θ_1 executes point sampling and collision detection while the second thread Θ_2 evaluates the hypotheses provided by Θ_1. Regarding efficiency, this parallelization approach is unfavorable because it either requires a significant number of context switches if $T > C$ or involves idle times if $T = C$. The latter aspect is not negligible because if no hypotheses were generated, Θ_2 either is a passive thread or assists Θ_1 to fill the relation tables (in the remainder of this thesis, the terms *hash table* and *relation table* are used synonymously). However, this stalls the system significantly because Θ_2 frequently switches the algorithmic context. Another option of parallelization is to let $T \geq C$ threads perform every step of the algorithm. The implemented approach to concurrent RANSAM features $T \geq C$ threads, each thread performing point sampling, collision detection, and hypothesis checking. Fig. 4.3 shows its structure. The left part of the figure displays the two relation tables R_A and R_B and two groups of threads entering point triples. The right part depicts two 3D fragments of the iRP robotics laboratory that have to be matched. In contrast to thread $\Theta_{\tau+1}$, thread Θ_2 has detected a collision after writing a point triple to R_A. The corresponding hypothesis ${}^{A}\boldsymbol{H}(\Theta_2)_B$ is then evaluated by Θ_2. Each hypothesis ${}^{A}\boldsymbol{H}(\Theta_i)_B$ computed by Θ_i is formally described as

$$ {}^{A}\boldsymbol{H}(\Theta_i)_B = {}^{\mathcal{W}}\boldsymbol{T}(\Theta_i)_B \cdot {}^{\mathcal{W}}\boldsymbol{T}(\Theta_i)_A{}^{-1} \tag{4.9} $$

representing a transformation between the two point sets.

After each completed hypothesis evaluation, a thread has to estimate whether the quality Ω_{curr} of the current hypothesis is higher than that of the best hypothesis and overwrites Ω_{best} as well as ${}^{A}\boldsymbol{H}_{B,best}$ if necessary. Write and read accesses to this critical section have to be synchronized to avoid inconsistencies. Frequent execution of the necessary synchronization primitives leads to noticeable overhead reducing the achievable speedup of parallel implementations. However, the more iterations are executed, overwriting Ω_{best} is likely to get a rare event because no better hypothesis might be calculated. The original RANSAM implementation draws points from the sets in an alternating manner. Assigning one thread to process the points of set A and a second one – running on a different core – to work on set B improves cache efficiency compared to the original

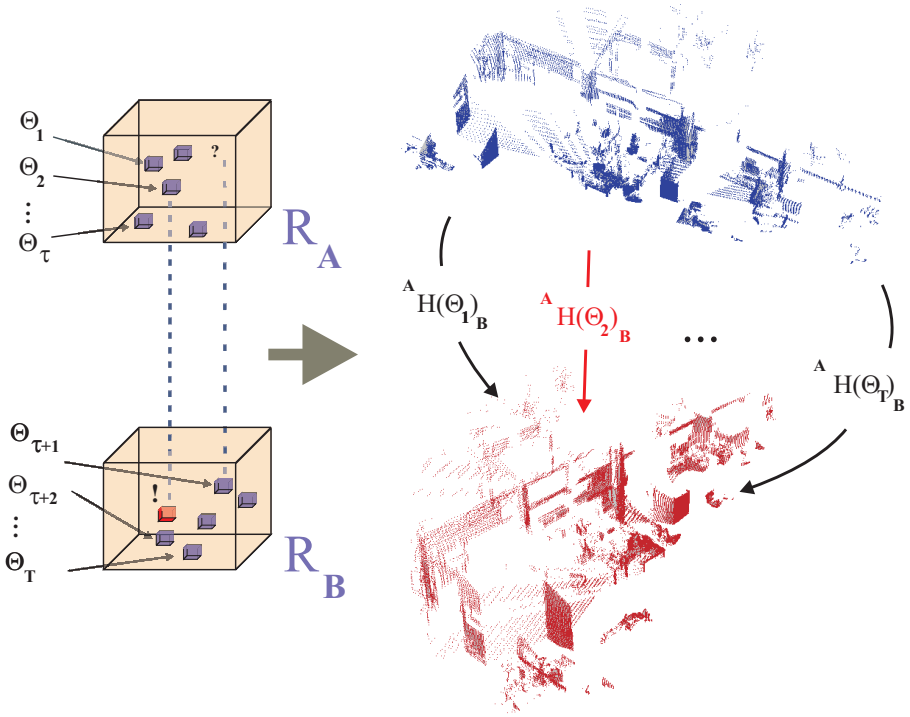


Figure 4.3.: Principle of pRANSAM. Each thread Θ_i fills one of the two relation tables, checks for collisions, and evaluates hypotheses ${}^{A}\boldsymbol{H}(\Theta_i)_B$.

implementation due to increased temporal and spatial locality. However, the impact of these effects heavily depends on the employed processor architecture and the operating system.

4.4.2. Optimal Relation Table Assignment to the Processor Cores

As stated above, an interesting and fundamental issue is whether there exists an optimal allocation of the tasks to the threads because the matching operation should provide a substantial solution as fast as possible for further processing. To this end, an estimate of the success probability of the pRANSAM algorithm is introduced, i.e. the probability that corresponding point triples in set A and set B are found. Based on the success probability, important characteristics of the algorithm can be derived. It will be proven that given T threads the success probability reaches its maximum if $\frac{T}{2}$ threads process point set A and $\frac{T}{2}$ threads process point set B.

An estimate of the success probability

Let A and B denote the point sets to be matched. P_I is the probability that after I iterations at least one correct correspondence between a point triple in A and a point triple in B has been found. Let

$$\overline{P_I} = 1 - P_I \tag{4.10}$$

denote the probability of the complementary event. P_i denotes the probability that at least one correct correspondence between a point triple in A and a point triple in B is found in iteration i. The probability of the complementary event, i.e. no correct correspondence is found in this iteration, is given by $\overline{P_i} = 1 - P_i$. $\overline{P_I}$ may thus be expressed as

$$\overline{P_I} = \prod_{i=1}^{I} \overline{P_i} \tag{4.11}$$

P_i may now be approximated as follows. Let T denote the number of threads of the parallel RANSAM algorithm and let τ denote the number of threads that draw point triples from set A; consequently $T - \tau$ threads process points of set B. $m = |A|$ and $n = |B|$ are the cardinalities of the sets A and B respectively. The total number of point triples in sets A and B amounts to $M_{tr} = \binom{m}{3}$ and $N_{tr} = \binom{n}{3}$ respectively. In the following $M_{tr}, N_{tr} \gg Ti$ is assumed. Furthermore, the entries in the relation tables are assumed to be uniformly distributed.

In each iteration, each of the τ threads randomly draws one point triple from set A and enters it into the relation table R_A, while each of the remaining $T-\tau$ threads draws one point triple from set B and enters it into the relation table R_B. After entering its point triple, each thread performs a collision check (cf. Eqn. (4.4)). The probability that one of the τ threads detects a collision in iteration i that indicates a correct correspondence may be approximated by

$$P_{i_A} \approx \frac{E\{p_B((T-\tau)\, i)\}\zeta_B}{M_{tr}} \tag{4.12}$$

whereas the probability that one of the $T - \tau$ threads is successful is approximately given by

$$P_{i_B} \approx \frac{E\{p_A(\tau i)\}\zeta_A}{N_{tr}}. \tag{4.13}$$

Here p_A and p_B are random variables and $E\{p_A(\tau i)\}$ and $E\{p_B((T-\tau)\, i)\}$ denote the expectation values of the number of points in the relation tables R_A and R_B after i iterations respectively. ζ_A and ζ_B consider properties of the sets A and B which are detailed below. As a result, the probability that none of the T threads finds a correct correspondence in iteration i is given by

$$\overline{P_i} \approx (1 - P_{i_A})^{\tau} (1 - P_{i_B})^{T-\tau} \tag{4.14}$$

assuming that P_{i_A} and P_{i_B} are constant during an iteration. Note that this is a simplifying assumption as P_{i_A} and P_{i_B} change negligibly during an iteration when a point triple is entered in the respective relation table.

If a new point triple that is entered into a relation table hashes to an already occupied slot, it overwrites the previously stored entry. Thus, the number of point triples k in a relation table after entering h triples must be $k \leq h$. The probability of k point triples in relation table R_A and R_B after entering h point triples is given by $p_A(h,k)$ and $p_B(h,k)$.

The actual sizes of the relation tables R_A and R_B are denoted by $\mathcal{M}$ and $\mathcal{N}$ respectively. Let $\mathcal{M}_e$ and $\mathcal{N}_e$ denote the *effective* size of the relation tables ($Ti \ll \mathcal{M}_e, \mathcal{N}_e \leq \mathcal{M}, \mathcal{N}$),

i.e. the number of slots that are actually occupied when entering the entire point sets A and B. $\mathcal{M}_e$ and $\mathcal{N}_e$ are directly affected by the geometrical structure of the point set A and B. If the bounding boxes limiting the geometrical extensions of A and B do not exceed the spatial extensions of the relation tables, then the relation tables are never completely filled with point triples.

$$p_A(h,k) = \frac{\binom{\mathcal{M}_e}{k}\binom{h}{k}\, k!\, k^{h-k}}{(\mathcal{M}_e)^h} \tag{4.15}$$

$$p_B(h,k) = \frac{\binom{\mathcal{N}_e}{k}\binom{h}{k}\, k!\, k^{h-k}}{(\mathcal{N}_e)^h} \tag{4.16}$$

The terms defining $p_A(h,k)$ and $p_B(h,k)$ are exemplified using equation 4.15. For simplicity, consider a one dimensional array of $\mathcal{M}_e$ slots. Then a ball is thrown h times and the probability is calculated that exactly $k \leq h$ slots of the array have been hit. The denominator counts the number of possibilities to choose h slots from a total of $\mathcal{M}_e$ slots. The nominator is more complex and needs further explanation. $\binom{\mathcal{M}_e}{k}$ is the number of possibilities to choose exactly k different slots from $\mathcal{M}_e$ slots. This must be multiplied by the number of possibilities to distribute k *successful* trials among h trials denoted by $\binom{h}{k}$. Given a concrete choice of k slots, there exists $k!$ possibilities to hit the slots since the order does not play a role. Finally, each of the remaining $h-k$ trials must hit one of the k slots. It does not matter which and how many different slots are hit, thus this number sums up to k^{h-k}.

Consequently, $p_A \sim p_A(h,k)$ and $p_B \sim p_B(h,k)$. The desired expectation values are given by

$$E\{p_A(h)\} = \sum_{j=1}^{h} p_A(h,j)\; j \tag{4.17}$$

$$E\{p_B(h)\} = \sum_{j=1}^{h} p_B(h,j)\; j \tag{4.18}$$

These equations, however, contain binomial coefficients, which renders them cumbersome for the analysis presented below. Therefore, equations (4.17–4.18) are substituted by the following approximations:

$$E'\{p_A(h)\} = \mathcal{M}_e \left(1 - \left(1 - \frac{1}{\mathcal{M}_e}\right)^h\right) \tag{4.19}$$

$$E'\{p_B(h)\} = \mathcal{N}_e \left(1 - \left(1 - \frac{1}{\mathcal{N}_e}\right)^h\right) \tag{4.20}$$

The idea of this approximation is to multiply $\mathcal{M}_e$ and $\mathcal{N}_e$ with the probability that each individual slot is hit at least one time during h iterations. Given $\mathcal{M}_e$ slots, a single one is chosen with probability $\frac{1}{\mathcal{M}_e}$. Thus, the same slot is not chosen with probability

$\left(1-\frac{1}{\mathcal{M}_e}\right)^h$ after h iterations and therefore the probability for the complementary event is $1-\left(1-\frac{1}{\mathcal{M}_e}\right)^h$.

These approximations consider each slot of the relation table independently. Eqn. (4.19) and (4.20) may be further simplified by approximating $(1-\frac{x}{y})^y$ by e^{-x} for small x and large y. Therefore,

$$E'\{p_A(h)\} \approx \mathcal{M}_e(1-e^{-h/\mathcal{M}_e}) \tag{4.21}$$
$$E'\{p_B(h)\} \approx \mathcal{N}_e(1-e^{-h/\mathcal{N}_e}). \tag{4.22}$$

The terms $e^{-h/\mathcal{M}_e}$ and $e^{-h/\mathcal{N}_e}$ are approximated using the first three elements of their power series.

$$E'\{p_A(h)\} \approx \mathcal{M}_e\left(1-\left(1-\frac{h}{\mathcal{M}_e}+\frac{h^2}{2\mathcal{M}_e^2}\right)\right) = h-\frac{h^2}{2\mathcal{M}_e} \tag{4.23}$$
$$E'\{p_B(h)\} \approx \mathcal{N}_e\left(1-\left(1-\frac{h}{\mathcal{N}_e}+\frac{h^2}{2\mathcal{N}_e^2}\right)\right) = h-\frac{h^2}{2\mathcal{N}_e} \tag{4.24}$$

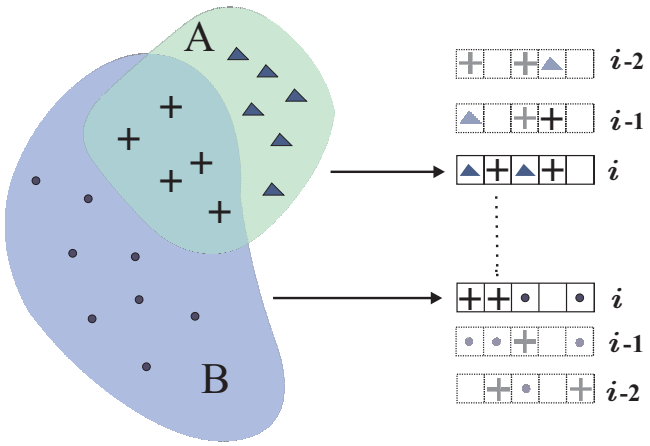


Figure 4.4.: Not each of the points belonging to set A may have a corresponding point in set B and vice versa, i.e. both point sets contain outliers. Consequently, each relation table may contain point triples for which it is impossible to compute a correct hypothesis.

The factors $\zeta_A, \zeta_B \in [0,1]$ consider an important characteristic of the point sets, viz. not each of the points contained in set A may have a corresponding point in set B and vice versa. ζ_A and ζ_B restrict the number of considered point triples in Eqn. (4.14) roughly to those which actually correspond to a triple in the other set.

$$\zeta_A = \frac{1}{M_{tr}}\binom{|A\cap B|}{3} \tag{4.25}$$
$$\zeta_B = \frac{1}{N_{tr}}\binom{|A\cap B|}{3} \tag{4.26}$$

This issue is exemplified in Fig. 4.4. The left part of the figure shows two intersecting point sets A and B; the right part shows a simplified view of the relation tables at iteration i, $i-1$, and $i-2$. The point sets A and B have common point triples indicated by + symbols. Only a point triple that is contained in both sets may yield a correct hypothesis. Therefore, regarding the relation tables at iteration i, at most one correct hypothesis may be derived (marked by the dotted line) whereas all other collisions in the depicted relation tables definitely lead to wrong hypotheses. This fact is considered by ζ_A and ζ_B, which reduce the success probability estimate accordingly.

Characteristics of the pRANSAM algorithm

Based on the equations estimating the success probability of pRANSAM, several important characteristics may be derived:

- In case of T threads, the maximum success probability is obtained by assigning $\frac{T}{2}$ threads to fill relation table R_A and $\frac{T}{2}$ threads to fill relation table R_B.
- The *theoretical* average number of matchings per time Σ_{match} increases linearly with the number of cores C in the system.
- Both average matching time and average matching quality strongly depend on the geometrical properties of the point sets. Matching a small segment of a map to a complete map should neither take less time nor yield a more precise result after a certain computation time as compared with matching two maps of similar sizes. To consider this, ζ_A, ζ_B, $\mathcal{M}_e$, and $\mathcal{N}_e$ need to be estimated.

The approximation of the success probability may now be used to prove characteristics of the pRANSAM algorithm. Let $f_A = \frac{\zeta_B}{M_{tr}}$ and $f_B = \frac{\zeta_A}{N_{tr}}$ be factors that do not depend on the number of iterations i. Then Eqn. (4.14) may be rewritten as

$$\overline{P_i} \approx \left(1 - \left(\nu i - \frac{\nu^2 i^2}{2\mathcal{N}_e}\right) f_A\right)^{\tau} \left(1 - \left(\tau i - \frac{\tau^2 i^2}{2\mathcal{M}_e}\right) f_B\right)^{\nu} \tag{4.27}$$

where $\nu = T - \tau$.

Theorem 1 (Optimal Distribution)**.** *Given $T \in \{k | k \in \mathbb{N} \wedge k \bmod 2 = 0\}$ threads and keeping the assumptions with respect to the point triple distribution stated above, the success probability P_I reaches its maximum if $\frac{T}{2}$ threads draw from point set A and $\frac{T}{2}$ threads draw from point set B.*

Proof. Employing $(1 - \frac{x}{n})^n \approx e^{-x}$ for small x, Eqn. (4.27) may be approximated by

$$\overline{P_i} \approx e^{\left(-\tau\nu i + \tau\frac{\nu^2 i^2}{2\mathcal{N}_e}\right) f_A} e^{\left(-\nu\tau i + \nu\frac{\tau^2 i^2}{2\mathcal{M}_e}\right) f_B} \tag{4.28}$$

Since the location of a minimum of a function is invariant with respect to logarithmization, the natural logarithm of Eqn. (4.28) is regarded in the following derivation.

Calculating the derivative of the logarithmized function yields

$$\begin{aligned}\frac{d\,log(\overline{P_i})}{d\tau} = & \underbrace{\frac{3}{2}i^2(\frac{f_A}{\mathcal{N}_e} - \frac{f_B}{\mathcal{M}_e})\tau^2}_{\approx 0} \\ + & \; 2i(f_A(1 - \underbrace{\frac{i\,T}{\mathcal{N}_e}}_{\approx 0}) + f_B(1 + \underbrace{\frac{i\,T}{2\mathcal{M}_e}}_{\approx 0}))\tau \\ - & \; iT(f_A + f_B - \underbrace{\frac{if_A\,T}{2\mathcal{N}_e}}_{\ll f_A})\end{aligned} \tag{4.29}$$

Equating to zero and neglecting small terms as indicated, leaves

$$2\tau\,(f_A + f_B) - T\,(f_A + f_B) = 0 \tag{4.30}$$

and hence $\tau = \frac{T}{2}$. As can be verified, the second derivative is positive at $\tau = \frac{T}{2}$. Since each factor of $\overline{P_I}$ (cf. Eqn. (4.11)) reaches its minimum at $\tau = \frac{T}{2}$, $\overline{P_I}$ reaches its minimum at $\tau = \frac{T}{2}$ and thus P_I its maximum. □

Therefore, an optimal allocation of tasks to threads exists that depends only on the number of employed threads T but not on any of the parameters ζ_A, ζ_B, $\mathcal{M}_e$, and $\mathcal{N}_e$.

Utilizing T processing nodes leads to T times the number of entries in the relation tables per time interval compared to the case of a single processing node. Keeping the assumptions stated above, the average matching time is thus reduced by a factor of T. A detailed complexity analysis of the original RANSAM method can be found in [6].

4.4.3. Real-Time Aspects

In several application areas real-time capability, i.e. low matching times are key requirements while accuracy is of secondary importance. Since the execution time per matching depends on the input data, the state of the random number generator, scheduling issues, etc., a time limit for matching operations has to be specified to employ pRANSAM in soft real-time applications. As pRANSAM provides a quality measure, each matching may be evaluated with respect to its plausibility. Occasional low-quality matchings may be compensated by employing probabilistic filters or context information. Generally, the average number of iterations and thus the average matching time may be reduced by modifying the contact criterion. Instead of a hard decision (cf. Eqn. (4.31)) whether points are in contact, an exponential function (cf. Eqn. (4.32)) for contact evaluation may be employed. Let $\Delta = |\boldsymbol{y}_j - \boldsymbol{x}'_i|$ denote the distance between a transformed point (cf. Eqn. (4.5)) and its closest point in set B. The hard contact criterion is defined as

$$contact_B^{hard}(\boldsymbol{x}_i) = \begin{cases} 1 & \Delta < \varepsilon_h \\ 0 & \Delta \geq \varepsilon_h \end{cases} \tag{4.31}$$

with the upper contact threshold ε_h. Additionally, for a smooth contact evaluation a lower threshold ϵ_s is defined which allows for a soft transition from *contact* to *no contact*:

$$contact_B^{soft}(\boldsymbol{x}_i) = e^{-c\cdot\Delta} \tag{4.32}$$

with $c = -\frac{2}{\varepsilon_h+\varepsilon_s} log\left(\frac{1}{2}\right)$. The constant c shifts $contact_B^{soft}(\boldsymbol{x}_i) = 0.5$ to the middle of ε_h and ε_s. For computational efficiency Eqn. 4.32 is approximated by a simple linear fuzzy-like contact criterion:

$$contact_B^{fuzzy}(\boldsymbol{x}_i) = \begin{cases} 1 & \Delta < \varepsilon_h \\ \frac{\varepsilon_s-\Delta}{\varepsilon_s-\varepsilon_h} & \varepsilon_h \leq \Delta < \varepsilon_s \\ 0 & \Delta \geq \varepsilon_s \end{cases} \tag{4.33}$$

As can be seen in the next section, the average matching time can be reduced significantly while at the same time preserving subjective matching quality.

4.5. Experimental Results

pRANSAM was implemented for the QNX real-time operating system (RTOS) and the Windows OS family to evaluate its performance on a broader basis. Although the other algorithms described in this work were implemented for a Windows OS, pRANSAM can be used in diverse applications, e.g. in scenarios where industrial robots are employed [3]. Those manipulators are often controlled via RTOS like QNX because hard time constraints have to be met. In this experimental section, the hardware and software setup is described. Subsequently, the scalability of pRANSAM is addressed and finally, experiments validating the characteristics derived in Sec. 4.4.2 are presented.

4.5.1. Hardware and Software Setup

A system equipped with an Intel Core 2 Quad Q9300 'Yorkfield' processor (256KBytes L1-Cache, 6144KBytes L2-Cache) running at $2500MHz$, an ASUS P5Q-E motherboard, and 2 GB of RAM with a clock rate of $1066MHz$ was employed in the experiments described in Sec. 4.5.2 and 4.5.4. The experimental results presented in Sec. 4.5.3 were obtained using the dual core system also employed for evaluation of the Map Matching Localization approach (cf. Sec. 5.7). The QNX version was tested with QNX Neutrino 6.3.2 (Multi-Core TDK 1.1.0) using the Intel C++ Compiler 8.1 for QNX Neutrino [24]. The Windows version was tested with Windows XP SP2 employing the Visual Studio 2003 tool chain.

The relation tables have been configured with 80 slots per axis. Their spatial dimensions depend on the scale of the point clouds to be matched. The parameters ε_h and ε_s were set to $\varepsilon_h = 0.5$ and $\varepsilon_s = 1.0$ which are standard settings applied to all scenarios discussed in this work. Their units depend on the resolution of the point clouds. Finally, a

quality threshold $\Omega_{thres} = 0.8$ was applied. In the following sections statistical data were obtained by executing 500 matching trials.

4.5.2. Scalability

To evaluate the scalability of the presented approach, matching times were measured for the original single core, dual core, and quad core version of the RANSAM algorithm. On QNX the threads were restricted to run on specific cores using bound multiprocessing (BMP) [25] to avoid thread migration which entails frequent cache thrashing. The theoretical speedup $S_C = \frac{t_1}{t_C}$ of the matching procedure is C, while the overall speedup depends on the speedup of the parallel kd-tree computation. In the remainder of this section, the first index of S represents the employed operating system, the second one denotes the number of threads, and the third one refers to the maximum permitted number of iterations I_{max}. Fig. 4.5 and 4.6 present average matching times for the QNX and the Windows version respectively.

The diagrams show average matching times for different numbers of utilized threads as well as maximum iterations ($I_{max,1} = 100,000$ and $I_{max,2} = 200,000$) when processing a 2D scene. The super linear speedups of the QNX version $S_{QNX,2,200K} = 2.4$ and the Windows version $S_{Win,2,200K} = 2.2$ may be attributed to the improved cache performance due to increased locality when one thread processes set A and the second one set B. The speedup $S_{QNX,4,200K} = 3.2$ of the quad core QNX version is conspicuously lower than the corresponding speedup of the Windows version $S_{Win,4,200K} = 4.0$. However, the single core and the dual core QNX versions outperform the corresponding Windows versions with respect to the average matching time. Comparing the quad core versions, the Windows version performs slightly better. As can be seen, utilizing more than $C = 4$ threads does not reduce matching time further. Regarding the fuzzy variant of the pRANSAM algorithm, average matching times for trials with $I_{max,1} = 100,000$ and $I_{max,2} = 200,000$ iterations do not differ significantly, which shows that the maximum number of iterations I_{max} is usually not reached. Compared to the non-fuzzy variant, a significant reduction of computational costs can be observed especially for $I_{max,2} = 200,000$. Table 4.1 depicts the average number of iterations using the fuzzy and the hard contact criterion respectively. As can be seen, when using the fuzzy contact criterion the number of iterations is approx. 37% less than the number of iterations when employing the hard contact criterion – depending on the input data and the parameters ε_h and ε_s.

Operating System	QNX		Windows	
Contact Criterion	Fuzzy	Hard	Fuzzy	Hard
Single Core	96840	155072	97315	152634
Quad Core	97774	155336	108749	161744

Table 4.1.: Average number of iterations ($I_{max} = 200,000$) using the fuzzy and the hard contact criterion respectively.

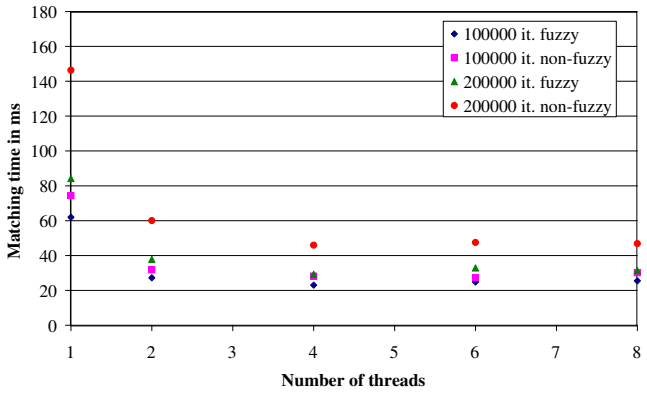


Figure 4.5.: Average matching times for the QNX version. Conspicuously super linear speedups $S_{QNX,2,200K} \approx 2.4$ were achieved for the dual core version but significantly reduced speedups $S_{QNX,4,200K} \approx 3.2$ were observed for the quad core version. When employing more than 4 threads on the quad core machine, the performance slightly deteriorates.

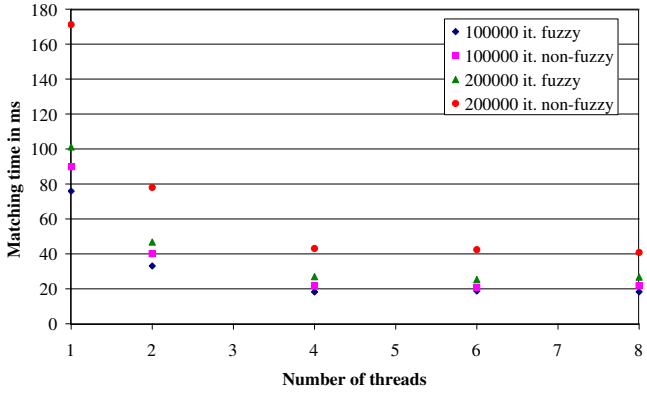


Figure 4.6.: Average matching times for the Windows version. Super linear and linear speedups were achieved for the dual core $S_{Win,2,200K} \approx 2.2$ and the quad core version $S_{Win,4,200K} \approx 4.0$ respectively. The speedup when employing more than 4 threads on the quad core machine is negligible.

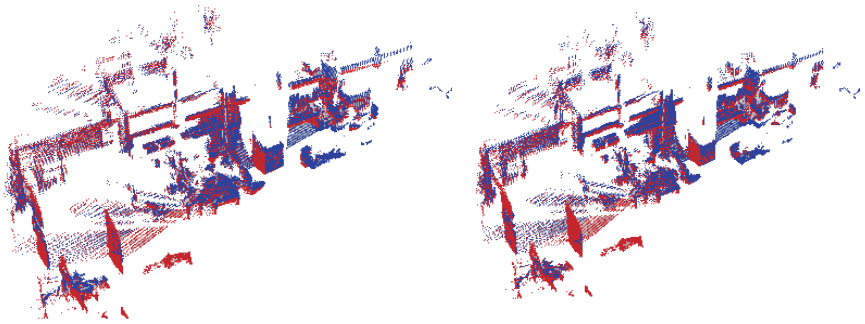


Figure 4.7.: Matching of the two scenes from the iRP laboratory (cf. Fig. 4.3). The hard contact criterion (cf. Eqn. (4.31)) was applied in the left figure with $\varepsilon_h = 0.5$, while the right figure depicts a matching result using the fuzzy contact criterion with $\varepsilon_s = 1.0$. The subjective matching quality does not deteriorate while a considerable speedup can be observed (cf. Figs. 4.5 and 4.6).

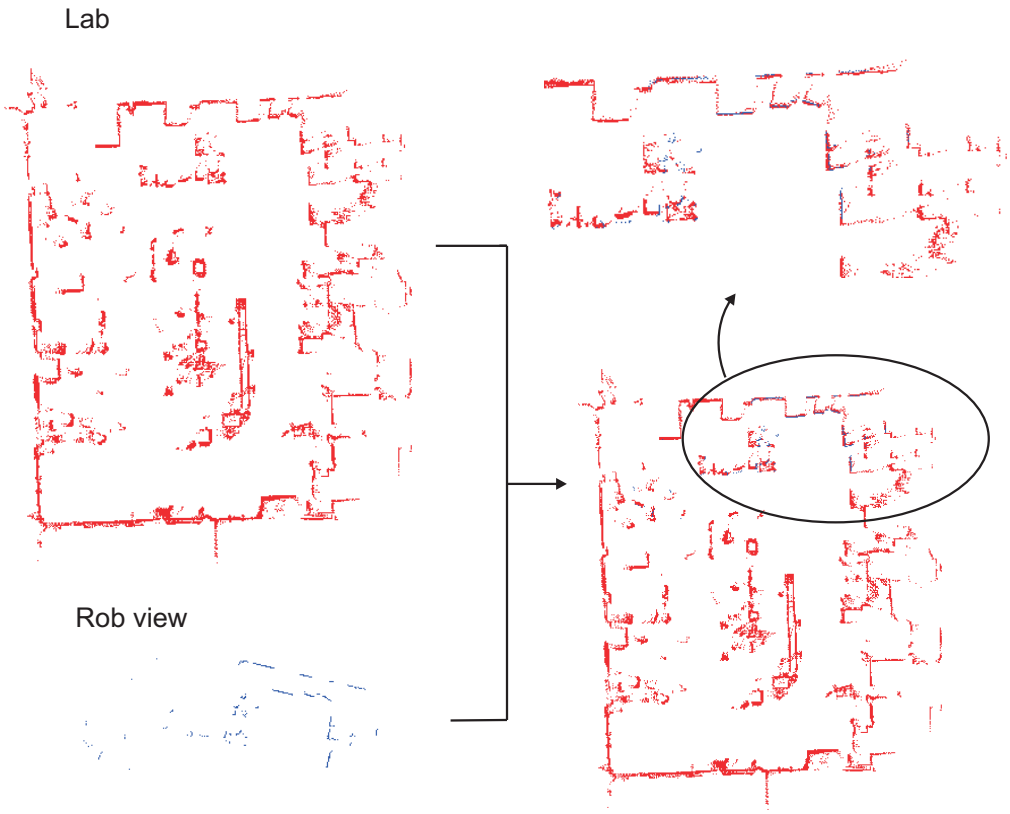


Figure 4.8.: Upper left: 2D map of the basement floor of the iRP laboratory. Lower left: A segment of the laboratory gathered with a SICK laser scanner mounted on one of our mobile robots. Right: Result of matching the robot view with the map.

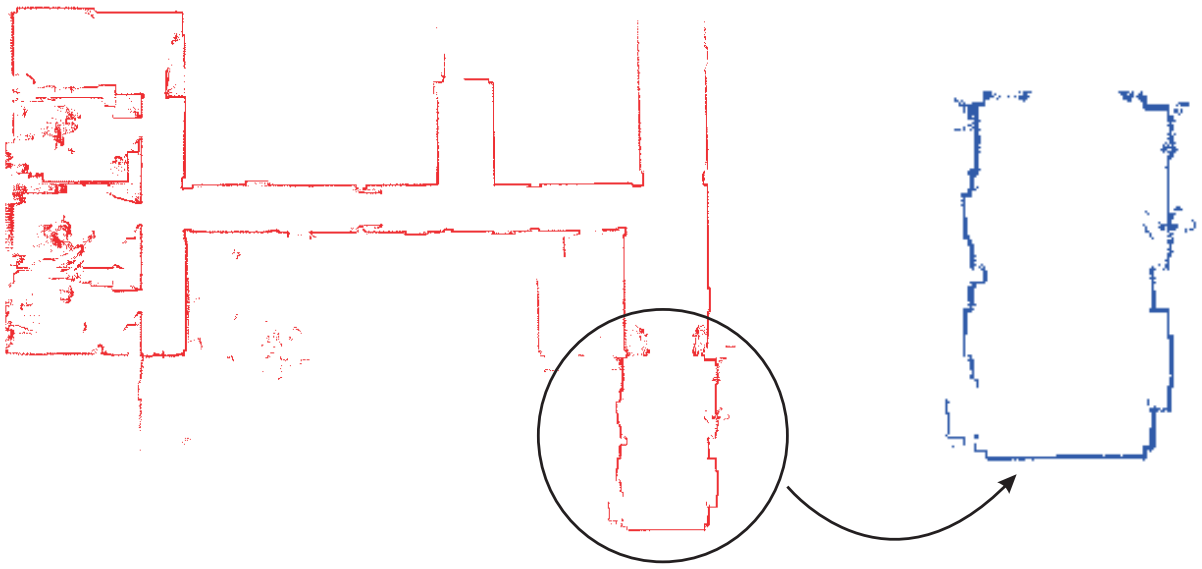


Figure 4.9.: a) Left: A 2D map of the ground floor of the iRP laboratory. Right: A small portion of the floor which is matched to the map. The cardinalities of both point sets are highly asymmetric.

4.5.3. Fuzzy Contact Criterion

The influence of the fuzzy contact criterion on the matching quality is investigated in this section. The experimental evaluation considers two aspects. First, the distances to the ground truth transformations were measured. Here, ground truth is expressed by a six dimensional vector $(t_x, t_y, t_z, \Theta_r, \Theta_p, \Theta_y)$ representing the true translation and the true orientation in rpy-angles (cf. App. B). The ground truth was generated by matching the different point clouds manually. The second issue was the computational gain, i.e. a faster matching time. The fuzzy contact criterion was introduced in order to save computation time while preserving an acceptable average matching quality at the same time. Consequently, it is necessary to investigate whether simply choosing a larger hard contact criterion (e.g. $\varepsilon_h = 1.0$) has the same (positive) effect. In this case the fuzzy contact criterion would be useless. As already stated in Sec. 4.5.1 the parameters ε_h and ε_s were set to $\varepsilon_h = 0.5$ and $\epsilon_s = 1.0$. Tab. 4.2 to 4.7 show the average matching errors and the average matching times for different I_{max} (compare Sec. 4.5.2). Three scenarios were studied, namely the 3D scene (cf. Fig. 4.7), the 2D scene depicted in Fig. 4.8, and a 2D scene of the ground floor of the iRP laboratory (cf. Fig. 4.9). Note, that the rotational error was calculated angle-wise. In the general case this is not correct since multiple representations of the same rotation are not considered. In other words the individual distances may be large while the overall rotations are very close to each other [26]. However, the second and the third scenario represent point clouds located on a plane. Thus, only the yaw angle is of importance. The errors of the 3D scene are very small. Therefore, the aforementioned problem can safely be ignored.

As Tab. 4.2 illustrates, using the fuzzy contact criterion in the 3D scenario does not yield a better average matching quality compared to a hard contact criterion with a higher threshold ($\varepsilon_h = 1.0$). Nevertheless, the fuzzy contact criterion results in the fastest computation time (cf. Tab. 4.3). This may be counter-intuitive but can be explained with more iterations during the extrapolation step when employing $\varepsilon_h = 1.0$. Therefore, the fuzzy contact criterion is preferable in this scenario.

The results of the basement floor of the iRP laboratory are depicted in Tab. 4.4 and 4.5. The hard contact criterion with $\varepsilon_h = 1.0$ yields a 23 percent faster computation time on average than the fuzzy contact criterion. In contrast, for $I_{max} > 50,000$ the matching error of the fuzzy contact criterion is slightly smaller than the matching error of the hard contact thresholds. Thus, the optimal strategy depends on the application.

The fuzzy contact criterion clearly outperforms the hard contact criterion with $\varepsilon_h = 1.0$ in the third scenario. Tab. 4.6 implies that $\varepsilon_h = 1.0$ is insufficient since the average matching errors are significant. The reason is that the third dataset has a coarser resolution than the other datasets. As a result, pRANSAM is very likely to generate false matches since the quality threshold is exceeded for many transformations which are far away from the correct one. The fuzzy contact criterion reaches a speedup of 31 percent (cf. Tab. 4.7) compared to the hard contact criterion with $\varepsilon_h = 0.5$ while being on the same level of matching quality.

I_{max}	**No Fuzzy** $\varepsilon_h = 0.5$	**Fuzzy**	**No Fuzzy** $\varepsilon_h = 1.0$
50 K	(0.62 0.40 1.11)	(0.53 0.33 1.00)	(0.68 0.37 0.87)
50 K	(0.03 0.06 0.01)	(0.03 0.05 0.01)	(0.03 0.05 0.02)
100 K	(0.31 0.27 0.59)	(0.34 0.22 0.67)	(0.33 0.23 0.67)
100 K	(0.02 0.03 0)	(0.01 0.03 0)	(0.01 0.03 0)
150 K	(0.27 0.19 0.48)	(0.22 0.17 0.48)	(0.27 0.18 0.49)
150 K	(0.01 0.03 0)	(0.01 0.02 0)	(0.01 0.02 0)
200 K	(0.24 0.15 0.40)	(0.24 0.13 0.42)	(0.25 0.16 0.45)
200 K	(0.01 0.02 0)	(0 0.02 0)	(0 0.02 0)

Table 4.2.: The matching error of the 3D scene for different I_{max} in x[dm], y[dm], z[dm], Θ_r[rad], Θ_p[rad], Θ_y[rad].

I_{max}	**No Fuzzy** $\varepsilon_h = 0.5$	**Fuzzy**	**No Fuzzy** $\varepsilon_h = 1.0$
50 K	210.305	155.805	207.216
100 K	295.664	234.659	293.961
150 K	393.492	314.726	361.201
200 K	512.638	409.669	424.931

Table 4.3.: The matching time of the 3D scene for different I_{max} in [msec].

I_{max}	**No Fuzzy** $\varepsilon_h = 0.5$	**Fuzzy**	**No Fuzzy** $\varepsilon_h = 1.0$
50 K	(11.42 33.77 4.60)	(13.63 32.16 4.20)	(9.24 27.37 3.20)
50 K	(0.72 0 0.59)	(0.65 0 0.62)	(0.50 0 0.48)
100 K	(1.26 1.86 0.40)	(0.61 1.28 0)	(1.76 2.31 0.40)
100 K	(0.06 0 0.03)	(0 0 0.04)	(0.06 0 0.05)
150 K	(0.28 0.28 0)	(0.27 0.28 0)	(0.42 0.31 0)
150 K	(0 0 0.01)	(0 0 0.01)	(0 0 0.01)
200 K	(0.24 0.24 0)	(0.22 0.26 0)	(0.38 0.31 0)
200 K	(0 0 0)	(0 0 0)	(0 0 0.01)

Table 4.4.: The matching error of the basement floor of the iRP laboratory for different I_{max} in x[dm], y[dm], z[dm], Θ_r[rad], Θ_p[rad], Θ_y[rad]. Compared to $\varepsilon_h = 1.0$, slightly better results are obtained when using the fuzzy contact criterion.

I_{max}	**No Fuzzy** $\varepsilon_h = 0.5$	**Fuzzy**	**No Fuzzy** $\varepsilon_h = 1.0$
50 K	36.093	29.731	28.582
100 K	77.535	57.704	42.057
150 K	118.451	66.261	48.353
200 K	165.842	78.818	51.901

Table 4.5.: The matching time of the basement floor of the iRP laboratory for different I_{max} in [msec].

I_{max}	**No Fuzzy** $\varepsilon_h = 0.5$	**Fuzzy**	**No Fuzzy** $\varepsilon_h = 1.0$
50 K	(0.40 2.04 0)	(9.71 2.56 1.2)	(48.09 29.54 7.00)
50 K	(0 0 0.02)	(0.18 0 0.19)	(1.09 0 1.11)
100 K	(0.35 1.86 0)	(4.08 2.24 0.40)	(46.20 32.59 7.40)
100 K	(0 0 0.01)	(0.06 0 0.08)	(1.16 0 1.11)
150 K	(0.46 2.46 0)	(2.31 2.26 0.2)	(45.48 32.42 8.80)
150 K	(0 0 0.02)	(0.03 0 0.05)	(1.38 0 1.06)
200 K	(0.38 1.95 0)	(2.32 2.54 0.20)	(44.96 30.39 6.40)
200 K	(0 0 0.02)	(0.03 0 0.05)	(1.00 0 1.06)

Table 4.6.: The matching error of the ground floor of the iRP laboratory for different I_{max} in x[dm], y[dm], z[dm], Θ_r[rad], Θ_p[rad], Θ_y[rad]. Compared to $\varepsilon_h = 1.0$, the fuzzy contact criterion yields significant better results.

I_{max}	**No Fuzzy** $\varepsilon_h = 0.5$	**Fuzzy**	**No Fuzzy** $\varepsilon_h = 1.0$
50 K	20.844	13.626	8.548
100 K	19.657	15.034	8.729
150 K	20.232	12.477	9.071
200 K	18.520	12.637	8.814

Table 4.7.: The matching time of the ground floor of the iRP laboratory for different I_{max} in [msec].

4.5.4. Validation of Characteristics

The characteristics derived in Sec. 4.4.2 were validated experimentally. To this end, the same scenarios as in Sec. 4.5.3 were studied and results are presented in this section. The data sets are abbreviated as scene 1, 2, and 3. Figs. 4.10 and 4.11 show the average matching quality and average number of collisions for all three data sets. The average matching time and the average number of iterations are depicted in Fig. 4.12 and 4.13. Results for both the fuzzy and the non-fuzzy contact criterion are presented. In scene 1, two point sets of roughly equal cardinalities and similar contents are matched. In contrast, the map of scene 2 (cf. Fig. 4.8) consists of approximately 20,000 points. A single laser scan gathered with a SICK laser scanner is matched which constitutes merely a small segment of the map. The scan contains only 361 points. Thus the cardinality ratio of both points sets is nearly 2 percent. In scene 3, a small set of laser observations (cf. Fig. 4.9 right) is matched to the map (cf. Fig. 4.9 left) which consists of roughly 16,000 points. The observation set contains around 1,500 points which yields a cardinality ratio of 9 percent.

For each number of permitted maximum iterations I_{max} from $5,000$ to $200,000$(increment: $5,000$), 500 matching trials were executed to determine the average matching quality and the average number of collisions.

In Sec. 4.4.2 an optimal distribution of processing nodes was derived. As Theorem 1 claims, the probability of finding a correct correspondence reaches its maximum if half

the nodes process set A and half work on set B. Fig 4.11 shows the highest number of collisions, which is closely related to P_I, (for both the fuzzy and the non-fuzzy contact criterion) if $\tau = \frac{T}{2} = 2$.

Fig. 4.10a and 4.10b depict the average matching quality of scene 1 plotted against the number of iterations. The average number of collisions is shown in Figs. 4.11a and 4.11b. Both point sets contain roughly $43,000$ points. As can be seen, the average matching quality as well as the average number of collisions converge at a lower number of iterations if the fuzzy contact criterion is used. Moreover, both the average matching quality and the number of collisions reach their maximum for the processor distribution according to Theorem 1, i.e. $\tau = 2$, regardless of the number of iterations. The average matching quality and the average number of collisions for $\tau = 1$ and $\tau = 3$ are roughly identical since both sets have approximately the same cardinality and content. Regarding the plots for the fuzzy contact criterion, a significantly lower number of collisions can be observed compared to the plots for the hard contact criterion since less iterations are required here. Fig. 4.12a and 4.12b highlight the average matching time plotted against the number of iterations. The average number of iterations are depicted in Fig. 4.13a and 4.13b. Obviously, the highest matching time is required for $\tau = 2$. This is comprehensible since the highest number of collisions occur for $\tau = 2$ and each collision causes a hypotheses evaluation which is a computationally expensive operation. On the contrary, $\tau = 2$ requires the lowest number of iterations until the matching threshold is exceeded because $\overline{P_i}$ (cf. Eqn. 4.27) reaches its minimum here.

Fig. 4.10c and 4.10d depict the average matching quality of scene 2 plotted against the number of iterations. The average number of collisions is shown in Figs. 4.11c and 4.11d. In contrast to scene 1, the data sets used in this figure are highly asymmetric with respect to their cardinality, one set containing $20,000$ points and the other one merely 360. Compared to Fig. 4.10a and 4.10b, the average matching quality requires more iterations to converge which is due to the characteristics of the data sets, i.e. a small segment of a map is mapped to a complete map. Hence ζ_B is very small. Furthermore, a clear difference between the results for $\tau = 1$ and $\tau = 3$ can be observed. If $\tau = 3$ only one thread processes the map (set B) and therefore very few collisions are obtained resulting in a low average matching quality. Due to the characteristics of the sets mentioned above, the results for $\tau = 1$ are close to those for $\tau = 2$. Nevertheless, Theorem 1 holds and the maximum average matching quality is observed for $\tau = \frac{T}{2} = 2$. Fig. 4.12c and 4.12d highlight the average matching time plotted against the number of iterations. The average number of iterations are depicted in Fig. 4.13c and 4.13d. In this scenario, $\tau = 1$ consumes the highest computation time. Furthermore, more collisions occur for $\tau = 2$. The reason for this is not clear but it is very likely that more time is required for $\tau = 1$ because the number of collisions are very close to $\tau = 2$ and $\tau = 2$ converges at a lower number of iterations for both the fuzzy and no fuzzy contact criterion.

Fig. 4.10e and 4.10f depict the average matching quality of scene 3 plotted against the number of iterations. The average number of collisions is shown in Fig. 4.11e and 4.11f. In contrast to scene 1 and 2, pRANSAM converges at roughly 50,000 to 60,000 iterations.

Applying more iterations does not increase the average matching quality, even for $\tau = 3$ where three threads fill the relation table of the smaller point set. It is obvious that Theorem 1 is satisfied again because the best average matching quality is obtained for $\tau = 2$ when the maximum number of iterations is less than 50,000. The result for more than 50,000 iterations does not contradict Theorem 1 because at this point the probability of finding a good match is high enough even for $\tau = 3$. $\tau = 2$ also yields the highest number of collisions. Fig. 4.12e and 4.12f highlight the average matching time plotted against the number of iterations. The average number of iterations required to exceed the matching threshold are depicted in Fig. 4.13e and 4.13f. These curves perfectly fit to the derived theory. The discrepancy of both the average matching time and the average number of iterations is significant for $\tau = 2$ versus $\tau = 1$ and $\tau = 3$ which implies that the probability of finding a good match is considerable higher for $\tau = 2$. The conspicuously fast convergence is due to another scale of scene 3 versus scene 2. Here, a larger scale of 3 decimeter was chosen for the points compared to a scale of 1 decimeter for scene 2. Hence, the points are much more densely located to each other and fit better to the spatial resolution of the relation tables. Consequently, a better solution can be generated employing less iterations. A comparison of the curves of scene 2 and 3 implicitly revealed another characteristic of pRANSAM which is not modeled in Sec. 4.4.2. The larger the geometrical dimensions of a point set the more time is necessary to sample the point triples which are stored in the relation table. To be more precise, if point set A has geometrical properties exceeding the spatial extension of the relation table and the other has not, only relation table R_A is filled quickly with point triples. Simultaneously, the thread(s) sampling from B consume much time to generate point triples and thus after the maximum number of iterations has been exceeded, R_B does not contain many entries. As a result, the average matching quality is decreased and more iterations are necessary for a good match. This characteristic might also lead to a violation of Theorem 1 if more than $\frac{T}{2}$ threads are required to produce a suitable number of point triples.

4.6. Conclusion

In this chapter, a parallelized approach to random sample matching (pRANSAM) was presented. Registering point triples to the relation tables is distributed among several threads where each thread performs point sampling, collision detection and hypothesis evaluation. pRANSAM features a roughly linear to super linear speedup compared to the conventional approach as shown by the experiments. The possible speedup highly depends on the number of processing nodes and the operating system. The computational costs of pRANSAM can be decreased by employing a fuzzy contact evaluation function instead of a hard contact criterion while maintaining subjective matching quality. Regarding the hardware set-up, matching times for 2D laser scans of less than 20 ms were achieved. Therefore, pRANSAM is an interesting approach for applications requiring a high number of matching operations per time. Furthermore, interesting characteristics of pRANSAM were derived. It was proven that an optimal allocation

of the tasks to threads exists. Given T threads, the probability of finding a correct point to point correspondence reaches its maximum if $\frac{T}{2}$ threads process the first point set and $\frac{T}{2}$ threads work on the second point set. Thus, the optimal thread allocation does not depend on the geometrical properties of the point sets like cardinality etc. and therefore - for practical applications - it is easy to configure the thread allocation to guarantee a maximal success probability. Extensive experiments showed the correctness of the derived characteristics. Based on the experimental results, pRANSAM may facilitate real-time pose estimation or object recognition using laser scanners, tactile sensors, etc. in assembly planning or service robotics applications. Moreover, pRANSAM may considerably enhance localization and SLAM approaches in mobile robotics.

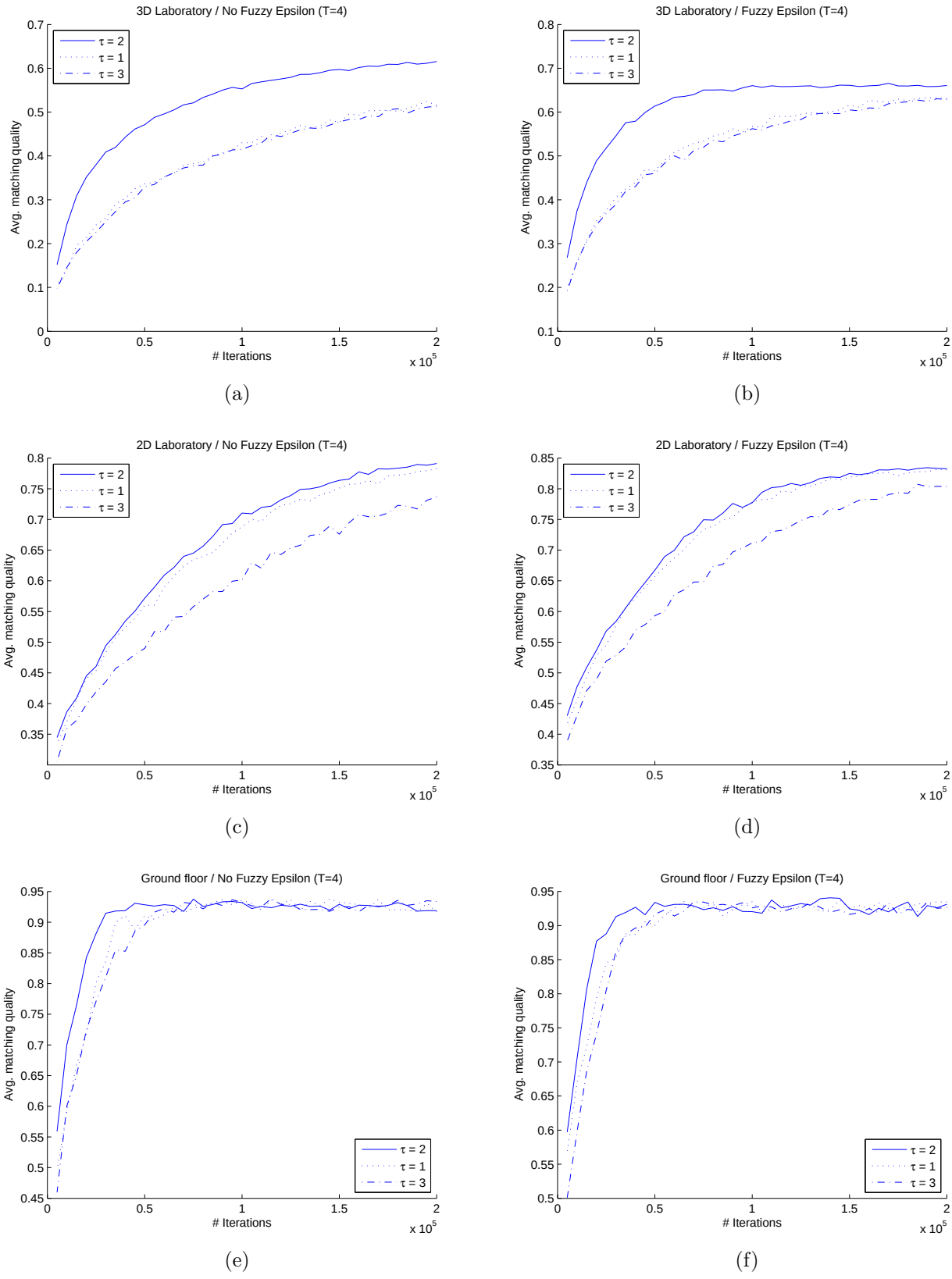


Figure 4.10.: Average matching quality plotted against I_{max}. a) Scene 1, no fuzzy contact criterion. b) Scene 1, fuzzy contact criterion. c) Scene 2, no fuzzy contact criterion. d) Scene 2, fuzzy contact criterion. e) Scene 3, no fuzzy contact criterion. f) Scene 3, fuzzy contact criterion.

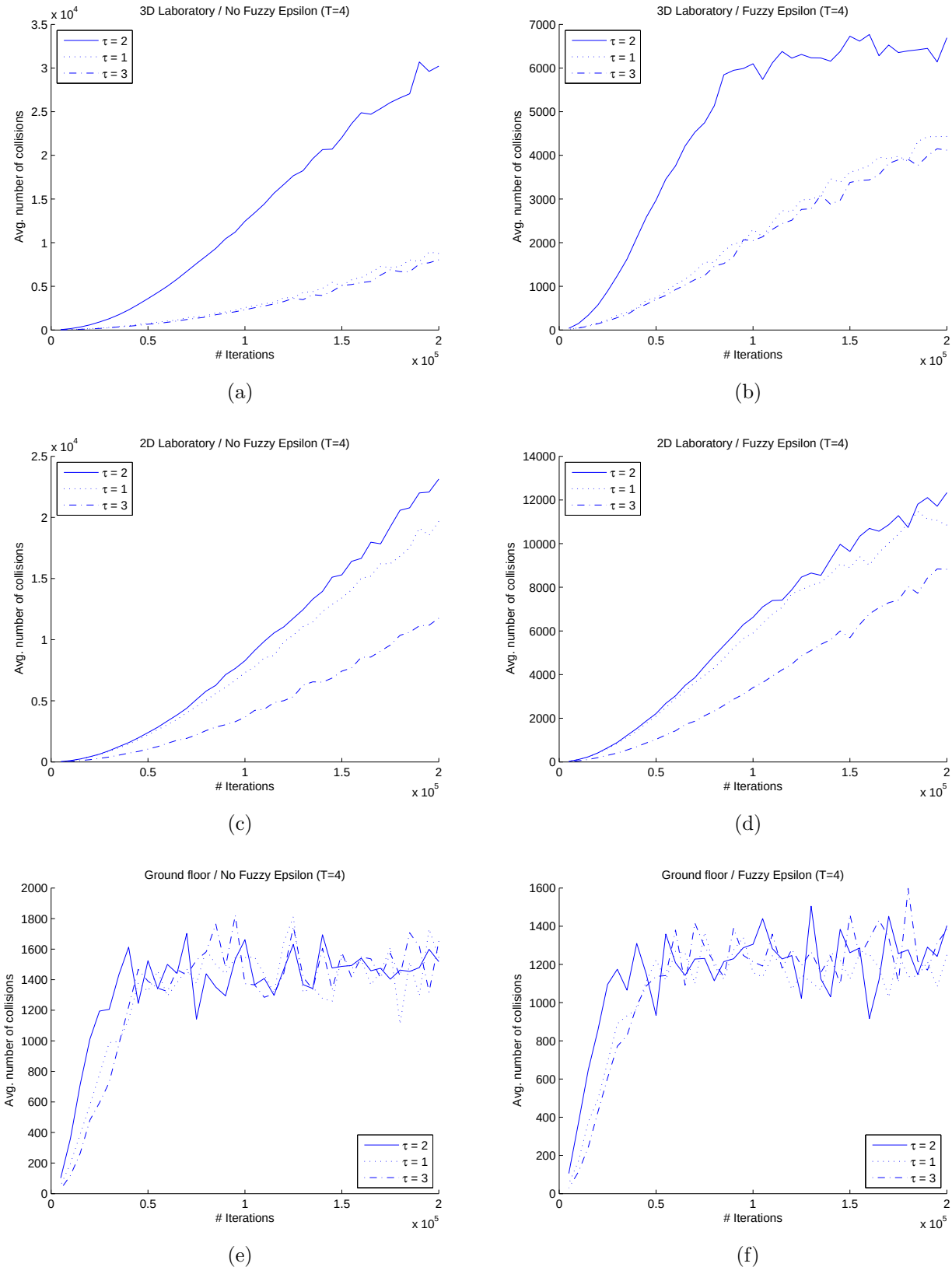


Figure 4.11.: Average number of collisions plotted against I_{max}. a) Scene 1, no fuzzy contact criterion. b) Scene 1, fuzzy contact criterion. c) Scene 2, no fuzzy contact criterion. d) Scene 2, fuzzy contact criterion. e) Scene 3, no fuzzy contact criterion. f) Scene 3, fuzzy contact criterion.

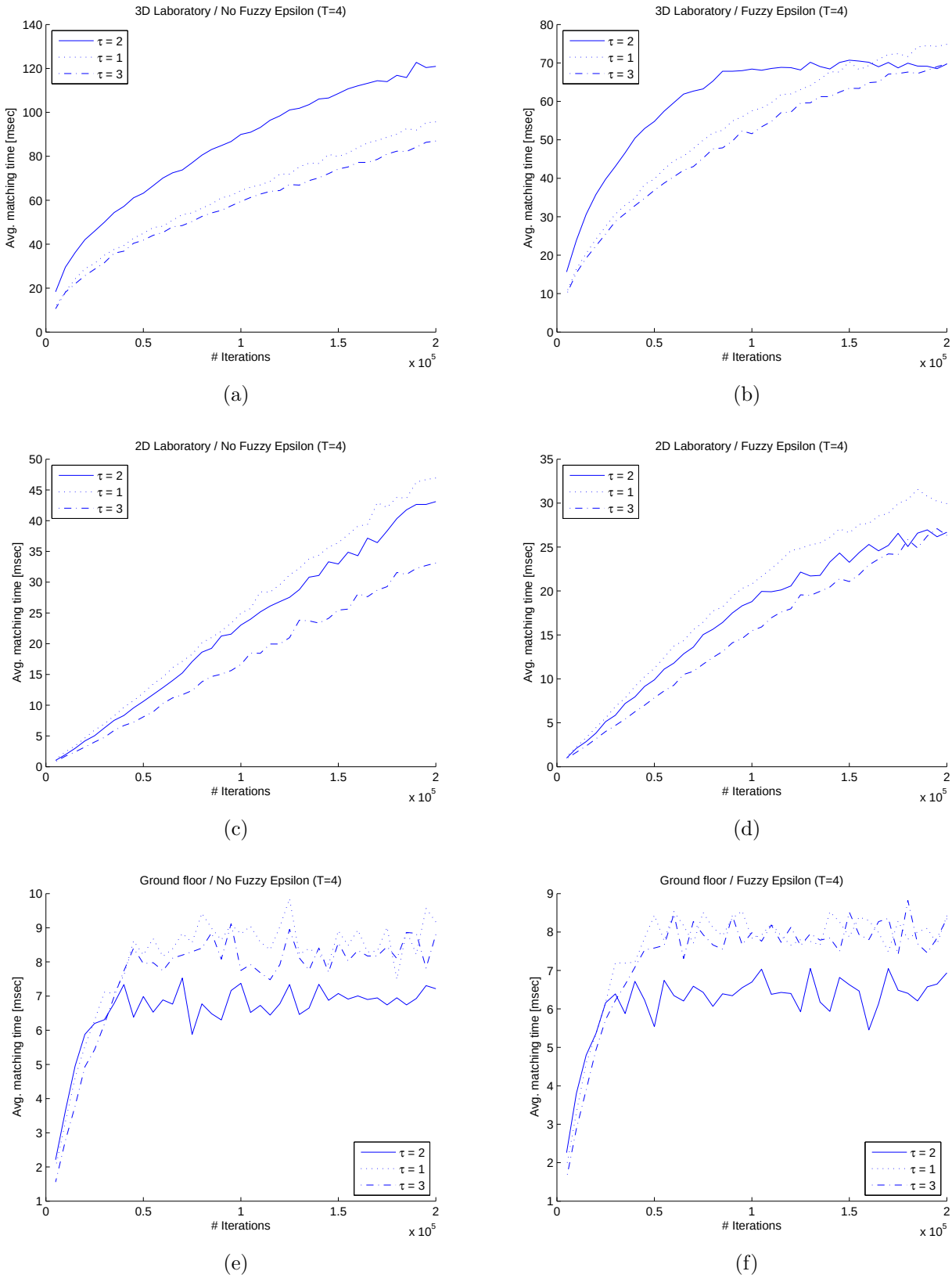


Figure 4.12.: Average matching time plotted against I_{max}. a) Scene 1, no fuzzy contact criterion. b) Scene 1, fuzzy contact criterion. c) Scene 2, no fuzzy contact criterion. d) Scene 2, fuzzy contact criterion. e) Scene 3, no fuzzy contact criterion. f) Scene 3, fuzzy contact criterion.

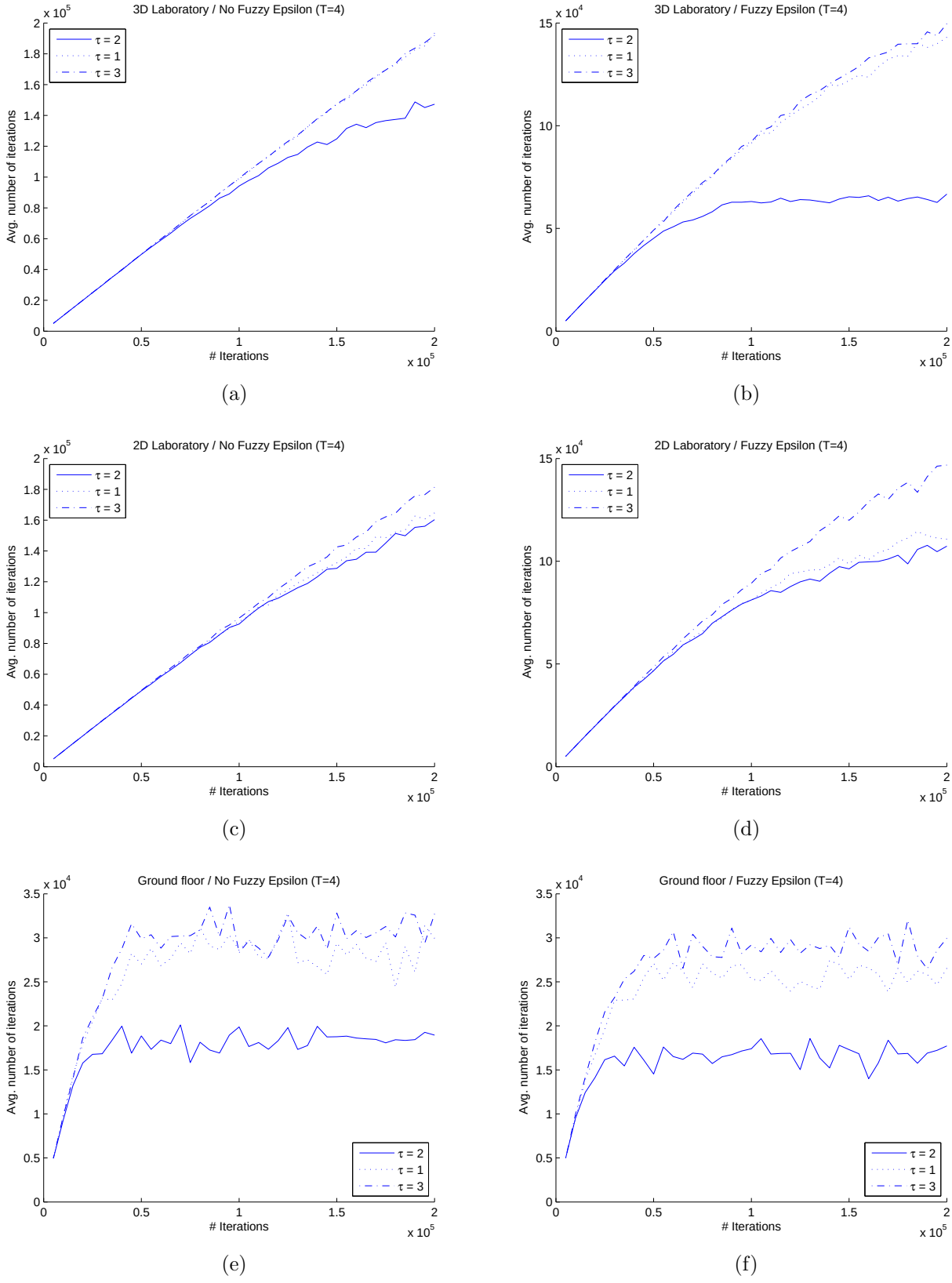


Figure 4.13.: Average number of required iterations plotted against I_{max}. a) Scene 1, no fuzzy contact criterion. b) Scene 1, fuzzy contact criterion. c) Scene 2, no fuzzy contact criterion. d) Scene 2, fuzzy contact criterion. e) Scene 3, no fuzzy contact criterion. f) Scene 3, fuzzy contact criterion.

References

[1] S. Petchartee and G. G. Monkman. Contact identification using tactile arrays. In *IEEE International Symposium on Signal Processing and Information Technology*, pages 1105 – 1110, December 2007.

[2] S. Haidacher and G. Hirzinger. Estimating finger contact location and object pose from contact measurements in 3d grasping. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 2, pages 1805 – 1810, September 2003.

[3] D. Kubus, R. Iser, S. Winkelbach, and F. W. Wahl. Efficient parallel random sample matching for pose estimation, localization, and related problems. In T. Kroeger and F. W. Wahl, editors, *Advances in Robotics Research*, pages 239–250. Springer-Verlag, 2009.

[4] R. Westphal. *Sensor-Based Surgical Robotics: Contributions to Robot Assisted Fracture Reduction*. Fortschritte in der Robotik. Shaker, 2007.

[5] S. Winkelbach. *Das 3d-Puzzle-Problem*. Fortschritte in der Robotik. Shaker, 2006.

[6] S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *Pattern Recognition (DAGM 2006)*, pages 718–728, 2006.

[7] R. Iser, D. Kubus, and F. Wahl. An efficent parallel approach to random sample matching (pRANSAM). In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1199–1206, 2009.

[8] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[9] C. Langis, M-Greenspan, and G. Godin. The parallel iterative closest point algorithm. In *Proc. of IEEE International Conference on 3-D Digital Imaging and Modeling*, pages 195–202, 2001.

[10] A. Nuechter. Parallelization of scan matching for robotic 3d mapping. In *Proc. of the 3rd European Conf. on Mobile Robots (ECMR)*, pages 198–203, 2007.

[11] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[12] A. M. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 46(6):381–395, 1981.

[13] D. Nister. Preemptive ransac for live structure and motion estimation. In *Proc. of IEEE International Conference on Computer Vision*, volume 1, pages 199 –206, October 2003.

[14] O. Chum and J. Matas. Randomized ransac with td,d test. *Image and Vision Computing*, 22(10):837 – 842, 2004.

[15] O. Chum and J. Matas. Optimal randomized ransac. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, August 2008.

[16] D. Chapel. An effective bail-out test for ransac consensus scoring. In *Proc. of British Machine Vision Conference (BMVC)*, pages 629–638, 2005.

[17] K. Tanaka and E. Kondo. Towards constant-time robot localization in large dynamic environments. In *Proc. of IEEE Int. Conf. on Networking, Sensing and Control*, pages 113–118, 2006.

[18] S. Se, G. D. Lowe, and J. J. Little. Vision based localization and mapping for mobile robots. *IEEE Trans. on Robotics*, 21(3):364–375, 2005.

[19] S. Se, G. D. Lowe, and J. J. Little. Global localization using distinctive visual features. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 226–231, 2002.

[20] E. W. Weisstein. Birthday attack. from mathworld - a wolfram web resource. ⟨http://mathworld.wolfram.com/birthdayattack.html⟩ [date: 07/08/2009]. Internet, 2009.

[21] C. Koenig. Master thesis: Implementation of a tactile sensor system for estimating object parameters. Institut fuer Robotik und Prozessinformatik, Technische Universität Braunschweig, 2008.

[22] R. Iser, J. Spehr, S. Winkelbach, and F. Wahl. Mobile robot localization using the fast random sample matching approach. In *Proc. of Robotik 2008*, pages 163–166, 2008.

[23] R. Iser and F. Wahl. Building local metrical and global topological maps using efficient scan matching approaches. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1023–1030, 2008.

[24] Intel c++ compiler 8.1 for qnx neutrino rtos ⟨http:/www.openqnx.com/node/344⟩ [date: 09/15/2011]. Internet.

[25] Qnx documentation library, community ressources, multicore processing ⟨http://www.qnx.org/developers/docs/6.3.2/neutrino/sys_arch/smp.html⟩ [date: 09/15/2011]. Internet.

[26] J. J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *Proc. of IEEE International Conference on Robotics and Automation*, volume 4, pages 3993 – 3998, 2004.

Chapter 5

MML - Map Matching Localization

5.1. Introduction

In the previous chapters important means were introduced constituting basic capabilities for the localization framework proposed in the following sections. A reliable scan matcher is available (cf. Chap. 2) and RANSAM providing efficient point cloud registration was parallelized for further speed-up (cf. Chap. 4).

As mentioned before, the basic variant of RANSAM was successfully employed in different applications like surface registration [1], medical robotics [2] or place recognition [3]. It is a fundamental part of the DAVID laser scanner [4] which nowadays is well-known to many research groups. The parallelized variant of RANSAM described in Chap. 4 (pRANSAM) is applied in this localization framework.

In a previous publication [5], first results of a RANSAM-based global localization scheme were presented. A single observation was matched to the map of the robot's workspace. Of course, matching only a single observation to the map may be insufficient due to structural ambiguity. Especially in a environment with few features, one observation may be matched to many parts of the map. For example, Fig. 5.1 shows an artificial environment with bare corridors. Fig. 5.1a depicts an observation of a sensor with limited view (marked by the red lines). Obviously, it is impossible to derive a unique robot pose as indicated by the triangles highlighted in Fig. 5.1b. This example perfectly illustrates that a matching algorithm employed for mobile robot localization must be embedded into a more sophisticated framework in order to be able to gather information over time. Consequently, the next sections elucidate how incremental local map building and global point cloud matching can be fused and dexterously utilized yielding a unique and accurate pose determination. The novel approach is termed as Map Matching Localization (MML).

5.1.1. Principle of the Localization Scheme

To this end, a fundamental requirement of pRANSAM is to provide a set of hypotheses which facilitates to decide whether the current robot pose can be estimated precisely enough. For this purpose, it is not sufficient to abort the computation of hypotheses as soon as the threshold Ω_{thres} is exceeded (cf. Chap. 4.3). Thus, pRANSAM was slightly modified. Now the algorithm runs a fixed number of iterations and collects all hypotheses generated during the iterations which exceed the threshold. Afterwards, this set of hypotheses is the basis for further processing, e.g. clustering several densely located hypotheses.

Another requirement is that the robot needs to build a local map of its environment during the localization process. Hence, not only a single observation is matched to the global map but the whole local map consisting of all observations made so far. This means that a local map carrying enough information allows pRANSAM to generate a unique pose hypothesis. The localization principle is elucidated in Fig. 5.1a to 5.1f. Fig. 5.1a depicts the path which is followed by the robot during the localization procedure. It starts from the pose marked by the blue triangle and ends in the upper left part of the map. As mentioned above, the first observation shown in Fig. 5.1a does not localize the robot due to a complete absence of rich geometrical features. Therefore, many hypotheses are computed as depicted in Fig. 5.1b. Fig. 5.1c shows the local map after integrating several observations into the local map. This decreases the uncertainty significantly since only two possible robot poses are left (cf. Fig. 5.1d). Nevertheless, these two poses have an equal level of confidence and thus localization is still postponed.

However, the uncertainty of the pose collapses when the robot enters the upper left part of the map (cf. Fig. 5.1e and 5.1f). All other poses have become very unlikely because the corridor widens in the upper right area . Consequently, for every pose hypothesized in this region, only few local map points would be in contact with the global map.

As a summary, the localization algorithm can be subdivided into three algorithmic steps. First a scan matcher fuses the current observation with all other observations made previously and thus adds more information to the local map. Then the local map is matched with the world model which yields a set of hypotheses. Finally, the hypotheses need to be analyzed in order to rate the level of uncertainty. Although the overall idea is pretty simple, many problems emerge from a detailed analysis of the requirements:

- Since pRANSAM is run a fixed number of iterations, it will yield a bunch of hypotheses even if the robot pose can be uniquely determined. Thus, the computed hypotheses have to be preprocessed which also includes a rating strategy.
- A decision rule needs to be defined which determines when the true robot pose is isolated accurately enough.
- A fixed matching threshold Ω_{thres} is not very practicable because the workspace of the robot cannot be assumed to be static. Consequently, a dynamic threshold will adapt the hypotheses generation to the current environmental situation.

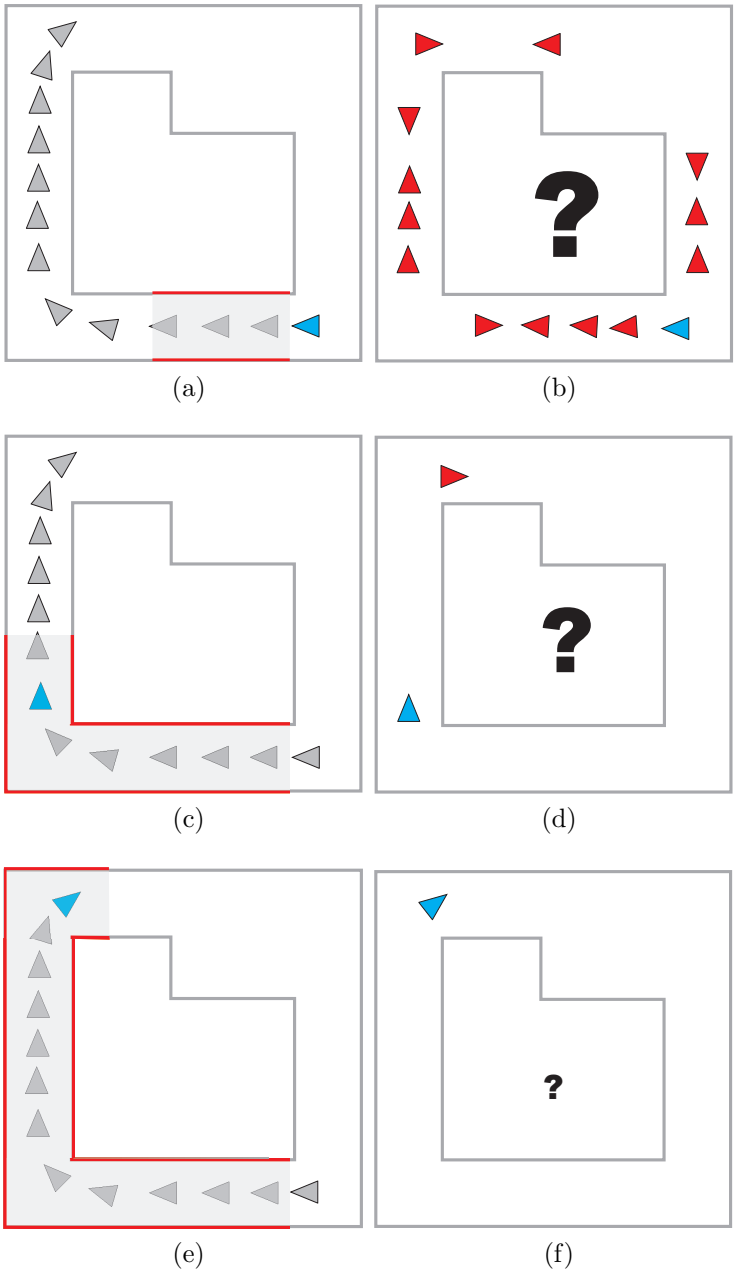


Figure 5.1.: Principle of the localization scheme. a) The entire path of the robot. b) Localization is still impossible due to the large number of hypotheses delivered by pRANSAM. c) The local map after a few observations. d) The uncertainty decreases significantly since only two possible robot poses are left. e) The local map at the end of the path. f) The algorithm terminates when the robot pose is isolated accurately enough.

- The local map is computed using the scan matcher described in Chap. 2. In workspaces of small or medium size a pure scan matcher may suffice to provide a local map which is accurate enough to localize the robot correctly. In larger environments where the robot needs to travel paths of considerable length, the local map will diverge. Hence, a strategy must be devised which is able to cope with that issue.
- The more the size of the local map increases the higher the matching time of pRANSAM. Depending on the application a linear rising computation time may render the algorithm impractical.
- If the robot moves to areas which are not registered in the global map, localization becomes impossible. Hence, it is necessary that the robot detects such a situation.

In the remainder of this chapter, related works and the above mentioned issues are discussed.

5.2. Related Works

In the last decades the global localization problem for mobile robots was discussed intensively in the literature. Thus, many localization techniques yielding impressive results exist nowadays. However, only RANSAC-based (cf. Chap. 4.2) localization methods are discussed in this section because these approaches are most relevant for the algorithm proposed in this chapter. A comprehensive overview of several localization paradigms and the respective literature is given in Chap. 1.

Se *et al.* [6] proposed a vision-based localization scheme by exploiting SIFT features [7]. The global map is represented by a database which comprises a set of landmarks. Each landmark is a SIFT feature and its corresponding position in the world. The localization works as follows: For each feature in the current camera image, its best match in the database is chosen as corresponding map feature. Then two alleged matches are selected randomly allowing to solve for the three unknown parameters of the robot pose. Afterwards, the hypothesis is evaluated by projecting each corresponding landmark back into the current image plane and calculating the discrepancy of expected and measured image position. Experimental results are presented investigating the characteristics of the algorithm in a 10 m $\times$ 10 m environment. The authors observed that a single observation is often insufficient for a reliable localization. Hence, a small local map is computed by rotating the robot a few degrees and then the local map is matched again.

Ju-Hong *et al.* [8] also employ a vision system for localization. A topological map on the global level is employed with nodes representing semantic places and arcs connecting the nodes. Each node comprises a metric map which contains 3D PCA-SIFT features. Localization is performed using a RANSAC scheme. In each iteration, three points are randomly selected which allow for computing the transformation matrix from the local robot frame to the global map.

Tanaka and Kondo [9, 10] use a RANSAC method in order to match a local map consisting of features to a global world model. Similar to the approach presented in this thesis the local map is built incrementally. Therefore the algorithm is termed incremental RANSAC (iRANSAC). It exploits a preemptive RANSAC scheme comparable to the one proposed by Nister [11] for map matching. The method maintains a feature and a hypothesis list, respectively. In each iteration a fixed number of feature-hypothesis pairs is chosen according to an order rule [11]. The order rule has been modified in order to optimize the competing demands of scoring as many hypotheses as possible and scoring preferred hypotheses by evaluating as many features as possible. Experiments demonstrate localization results in simulated large scale environments. Unfortunately, only very little is said about the accuracy and no localization constraint is discussed, i.e. when the algorithm accepts the best hypothesis as the true robot pose. Furthermore, no details are given concerning the hypotheses generation and thus this aspect remains an open issue. Ueda and Tanaka [12] applied iRANSAC to real-world scenarios by extracting generalized shape context (GSC) [13] features from scans of a laser range-finder. The ANN algorithm [14] is employed for solving the nearest neighbor problem during hypothesis evaluation. The authors emphasize the applicability of their technique to large scale environments but no indication is given about the geometrical extension of the workspace and the local map, respectively. Saeki *et al.* [15] further improved the iRANSAC scheme by introducing a technique named LSH-RANSAC. The overall idea remains the same but feature retrieval is based on locality sensitive hashing (LSH) [16] since LSH has proven to be significantly faster than ANN [17]. However, an enhanced LSH method is applied which is termed as Exact Euclidean LSH (E^2LSH) [17]. The experiments show interesting results since Saeki *et al.* [15] simultaneously build the global map using R mapper robots and employ a so called target robot to construct the local map. The mapper robots are distributed to several buildings and thus not only the pose of the target robot but also its workspace is unknown. The authors state that a scan matcher is used in order to compute the maps. However, nothing is said about the problem of map inconsistency since the proposed technique is devised for the application in large scale environments as claimed by the authors. Unfortunately, no total computation time is given which would be very helpful to asses the feasibility of the algorithm. The overall idea of matching local and global maps is very close to the approach presented in this thesis. The main difference of the proposals [9, 10, 15] discussed so far is that no features have to be extracted from the sensor data. Instead, metrical maps are matched to each other using a very efficient algorithm for point cloud registration. Consequently, the method introduced in this thesis is more flexible since it does not rely on the presence of discriminative features. Moreover, the above literature does not consider any constraint determining when localization is successful. Finally, the consistency of the local map is silently assumed although it is obvious that the errors accumulate over time no matter which scan matcher is employed.

5.3. Hypotheses Interpretation

5.3.1. Cluster Rating

From a theoretical point of view there should be only one unique robot pose left when enough information has been collected. But in practical situations pRANSAM is likely to generate a set of hypotheses which are located closely to each other instead of only one single hypothesis. This is due to the fact that $\varepsilon > 0$ (cf. Chap. 4.3) which allows for small translations and rotations around the *ideal* matching pose. Moreover, outliers are occasionally computed. An outlier is a hypothesis which is very unlikely but the matching quality Ω exceeds the current threshold Ω_{thres}. An example is depicted in Fig. 5.2a. Many hypotheses (red triangles) are located around the true robot pose while one outlier is generated (blue triangle) which results from an inappropriate Ω_{thres}, e.g. $\Omega_{thres} \approx 0.7$ is small enough to cause a suboptimal matching result as illustrated. A higher matching threshold prevents pRANSAM from computing such outliers. This matter is discussed in Sec. 5.3.3.

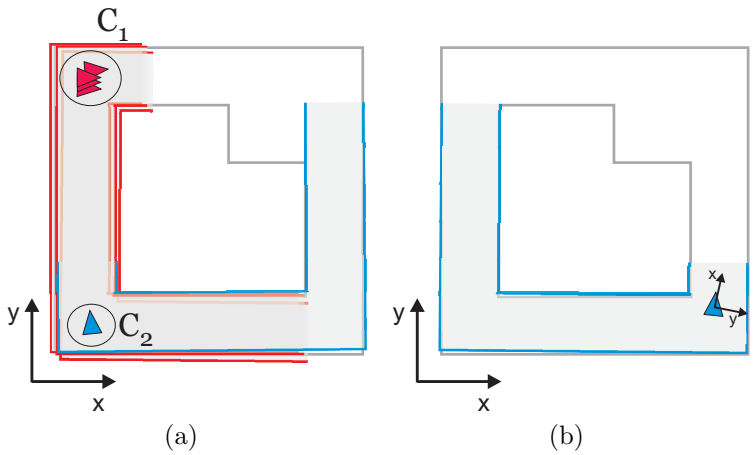


Figure 5.2.: a) The red triangles constitute hypotheses around the true robot location while the blue triangle represents an outlier. Based on these hypotheses, K-means provides two clusters $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. b) This hypothesis represents an impossible configuration since the z-axis points into the image plane.

Consequently, the behavior described above requires a hypotheses clustering routine. To this end, a hierarchical K-means clustering algorithm [18, 19] is used. Following the assumption that pRANSAM returns a set of hypotheses

$$\mathcal{H} = \left\{ {}^{A}\boldsymbol{H}_{B} \mid \Omega\left({}^{A}\boldsymbol{H}_{B}\right) \geq \Omega_{thres} \right\} \tag{5.1}$$

the K-means algorithm is initialized with one cluster $\boldsymbol{C}_1$ and the covariance matrix

$$\boldsymbol{\Sigma}_K = \begin{pmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{pmatrix} \tag{5.2}$$

of all poses belonging to this cluster is computed. The cluster is recursively subdivided if one of the elements is to large, i.e. if a threshold $\sigma^2_{trans,max}$ is exceeded. Afterwards, the same is done for the rotation around the z-axis. Here, a maximal variance $\sigma^2_{rot,max}$ may not be exceeded. The result is a set of clusters $\mathcal{C} = \{\boldsymbol{C}_i \mid i \in [1, M]\}$ with $M \leq \mid \mathcal{H} \mid$. An example is depicted in Fig. 5.2a. Hypotheses marked by the red triangles are located densely around each other while one outlier (blue triangle) has been generated. K-means produces two clusters $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. Here $\boldsymbol{C}_1$ contains all red poses and $\boldsymbol{C}_2$ contains the single blue pose. Note that some hypotheses can be excluded prior to clustering. Fig. 5.2b depicts an invalid configuration since the z-axis of the hypothesis points into the image plane while the z-axis of the world frame points out. Thus, the robot would be located *under* the map. As a result, a hypothesis $\boldsymbol{H}$ is valid if

$$\begin{aligned} \Theta_r(\boldsymbol{H}) &< \delta_r \\ \Theta_p(\boldsymbol{H}) &< \delta_p \end{aligned} \tag{5.3}$$

where δ_r and δ_p denote maximal rotation angles around the x- and y-axis. Otherwise, $\boldsymbol{H}$ is discarded. For more information concerning the computation of Θ_r and Θ_p, refer to App. B. Finally, a weight is assigned to each cluster which is employed as a measure of how reliable the cluster represents the current true robot pose:

$$\begin{aligned} r\left(\boldsymbol{C}_i\right) &= \alpha_1 \frac{\mid \boldsymbol{C}_i \mid}{\mid \mathcal{H} \mid} + \alpha_2 \left(1 - \frac{\sigma^2_{xx,i} + \sigma^2_{yy,i}}{2\sigma^2_{trans,max}}\right) \\ &\quad + \alpha_3 \left(1 - \frac{\sigma^2_{rr,i}}{\sigma^2_{rot,max}}\right) + \alpha_4 \frac{1}{\mid \boldsymbol{C}_i \mid} \sum_{j=1}^{|\boldsymbol{C}_i|} \Omega\left(c_{j,i}\right) \\ &= \alpha_1 w_1 + \alpha_2 w_2 + \alpha_3 w_3 + \alpha_4 w_4 \end{aligned} \tag{5.4}$$

with $\sum_{k=1}^{4} \alpha_k = 1$ and $r\left(\boldsymbol{C}_i\right) \in]0; 1]$. The first term w_1 of this equation rates how many hypotheses are assigned to the i-th cluster with respect to all hypotheses generated in the current localization step. Hence, the value of w_1 of the cluster $\boldsymbol{C}_2$ (cf. Fig. 5.2a) would be much smaller than w_1 of the cluster $\boldsymbol{C}_1$. The next two terms are assigned high values if the translational and rotational variances of the cluster are small. Finally, w_4 represents the average matching quality of the cluster. Empirical investigations showed that w_1 and w_4 are the most important terms. A faster and more reliable localization can often be achieved by a simple linear transformation $f : [\Omega_{thres}; 1] \rightarrow [0; 1]$ of w_4. Currently, $w_4 \in [\Omega_{thres}; 1]$. Hence,

$$f\left(w_4\right) = \frac{w_4 - \Omega_{thres}}{1 - \Omega_{thres}} \tag{5.5}$$

is applied to w_4, $f\left(w_4\right) \in [0; 1]$. In this way, its values are *stretched* and thus the effect

of small changes of the average quality of different clusters is increased. Consequently, Eqn. 5.4 is replaced by

$$r'(\boldsymbol{C}_i) = \alpha_1 w_1 + \alpha_2 w_2 + \alpha_3 w_3 + \alpha_4 f(w_4) \tag{5.6}$$

resulting in a localization criterion (described in 5.3.2) satisfied more often. The drawback is that the algorithm is more optimistic which raises the probability that the result is erroneous.

5.3.2. The Localization Criterion

The rating of equation 5.6 does not constitute an acceptable termination criterion because it is cumbersome to derive a feasible threshold for localization. For example, in Fig. 5.1d the clusters representing the two robot poses will receive equal weights. Additionally, each cluster will have a considerable weight close to 1 because all points of the local map have a corresponding point in the model. Hence, choosing the cluster with the highest weight will cause a wrong localization with a probability of fifty percent. In other scenarios even a unique cluster may get a low weight if the workspace is not static and the local map misses point-to-point correspondences. Thus, another rating $\overline{r(\boldsymbol{C}_i)}$ is introduced which is defined as

$$\overline{r(\boldsymbol{C}_i)} = \frac{r'(\boldsymbol{C}_i)}{\sum_{j=1}^{|\mathcal{C}|} r'(\boldsymbol{C}_j)}. \tag{5.7}$$

The rating $\overline{r(\boldsymbol{C}_i)}$ of a cluster $\boldsymbol{C}_i$ will get a significant value only if $r'(\boldsymbol{C}_i)$ is considerably higher than the ratings of all other clusters and thus ambiguous situations can be detected with a high reliability. For example, in Fig. 5.1d $\overline{r(\boldsymbol{C}_1)} \approx \overline{r(\boldsymbol{C}_2)} \approx 0.5$ while in Fig. 5.2a $\overline{r(\boldsymbol{C}_1)} >> \overline{r(\boldsymbol{C}_2)}$ and $\overline{r(\boldsymbol{C}_1)} >> 0.5$. Hence, one localization criterion is that $\overline{r(\boldsymbol{C}_i)} > \delta_r, \delta_r \in \mathbb{R}$. However, in challenging environments pRANSAM may generate only a single cluster although a unique pose isolation is still not possible yielding a false localization result. Consequently, another criterion is added which must be satisfied before the algorithm terminates. The pose of the current best cluster is compared to the *expected* robot pose which is calculated by applying a suitable odometry model to the best cluster of the previous iteration. To this end, let $\boldsymbol{x}_{prev} = (x_{prev}, y_{prev}, \theta_{prev})$ be the last best hypothesis. Furthermore, let $\boldsymbol{x}_r = (x_r, y_r, \theta_r)$ be the current relative displacement of the robot with respect to $\boldsymbol{x}_{prev}$. Then the expected active robot pose $\boldsymbol{x}_e$ is calculated as

$$\boldsymbol{x}_e = \begin{pmatrix} x_{prev} cos\theta_r - y_{prev} sin\theta_r + x_r \\ x_{prev} sin\theta_r + y_{prev} cos\theta_r + y_r \\ \theta_{prev} + \theta_r \end{pmatrix}. \tag{5.8}$$

Both $\boldsymbol{x}_{prev}$ and $\boldsymbol{x}_r$ have covariance matrices $\boldsymbol{P}_{prev}$ and $\boldsymbol{P}_r$, respectively. According to equations 2.14 and 2.15, the covariance P_e of $\boldsymbol{x}_e$ calculates to

$$\boldsymbol{P}_e = \boldsymbol{J}_{x_{prev}} \boldsymbol{P}_{prev} \boldsymbol{J}_{x_{prev}}^T + \boldsymbol{J}_{x_r} \boldsymbol{P}_r \boldsymbol{J}_{x_r}^T \tag{5.9}$$

with

$$\boldsymbol{J}_{x_{prev}} = \left.\frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{x}_{prev}}\right|_{x_{prev}, x_r}, \boldsymbol{J}_{x_r} = \left.\frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{x}_r}\right|_{x_{prev}, x_r}. \tag{5.10}$$

Let $(\boldsymbol{x}_h, \boldsymbol{P}_h)$ be the current best hypothesis generated by pRANSAM and its covariance. Now the objective is to measure the length of the vector $\boldsymbol{r}_h^e$ which represents the relative motion between $\boldsymbol{x}_e$ and $\boldsymbol{x}_h$. The length is weighted by its covariance $\boldsymbol{R}_h^e$ whose calculation is elucidated in the following. $\boldsymbol{r}_h^e$ is computed as

$$\boldsymbol{r}_h^e = (\ominus \boldsymbol{x}_e) \oplus \boldsymbol{x}_h \tag{5.11}$$

where $\ominus$ is the inverse operator and $\oplus$ is the concatenation of two motion vectors [20]. Consequently, analogous to [20], $\boldsymbol{R}_h^e$ is expressed as

$$\boldsymbol{R}_h^e = \boldsymbol{J}_{\ominus} \boldsymbol{P}_e \boldsymbol{J}_{\ominus}^T + \boldsymbol{J}_{\oplus} \boldsymbol{P}_h \boldsymbol{J}_{\oplus}^T \tag{5.12}$$

where

$$\boldsymbol{J}_{\ominus} = \left.\frac{\partial \boldsymbol{r}_h^e}{\partial \boldsymbol{x}_e}\right|_{x_e, x_h}, J_{\oplus} = \left.\frac{\partial \boldsymbol{r}_h^e}{\partial \boldsymbol{x}_h}\right|_{x_e, x_h}. \tag{5.13}$$

As a result, a second criterion states that the Mahalanobis distance

$$d^2\left(\boldsymbol{r}_h^e, \boldsymbol{R}_h^e\right) = \boldsymbol{r}_h^e \boldsymbol{R}_h^{e\,-1} \boldsymbol{r}_h^{e\,T} \leq {\xi_r}^2 \tag{5.14}$$

with a distance threshold ξ_r. Since it is well-known that the Mahalanobis distance follows a Chi-square distribution, ξ_r should be the p-quantile of a Chi-square distribution with a desired confidence level p. The idea of this criterion is exemplified in Fig. 5.3.

So far two conditions are defined which have to be met for localization. A third criterion affects the variance of the solutions with respect to Eqn. 5.14. For example, if a valid solution and a non-valid solution are computed in an alternating manner (which may happen since pRANSAM is a non-deterministic algorithm) it is also hard to decide whether the valid solution is reliable enough. To diminish this effect, the last τ_m distances are queued. Furthermore, one option would be to compute the mean and the variance of the stored distances. However, robustness to non-valid outliers is important

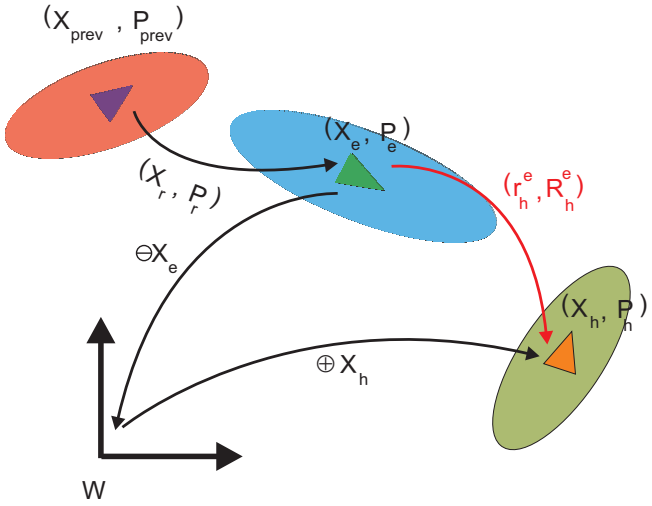


Figure 5.3.: Principle of the second condition. The best hypothesis or cluster $\boldsymbol{x}_{prev}$ from the previous iteration is moved using the current odometry input $\boldsymbol{x}_r$. This yields the expected robot pose $\boldsymbol{x}_e$ of the current iteration. The relative motion $\boldsymbol{r}_h^e$ between $\boldsymbol{x}_e$ and the current best hypothesis $\boldsymbol{x}_h$ generated by pRANSAM is highlighted by the red arrow. The length of this vector is an indicator for the confidence of the current solution.

and the mean and the variance are very sensitive to outliers. Therefore, the median of the last τ_m queued distances is computed first:

$$d_{med}^2 = median\left(\bigcup_{i=T-\tau_m}^{T} d^2\left(\boldsymbol{r}_h^e, \boldsymbol{R}_h^e\right)(i)\right) \tag{5.15}$$

Our measure of uncertainty is defined as the median of all distances to d_{med}^2:

$$\sigma_{med} = median\left(\bigcup_{i=T-\tau_m}^{T} \left|d^2\left(\boldsymbol{r}_h^e, \boldsymbol{R}_h^e\right)(i) - d_{med}^2\right|\right) \tag{5.16}$$

The third localization criterion is that $\sigma_{med} \leq \xi_{med}, \xi_{med} \in \mathbb{R}$. Although this measure is just a pure heuristic, it has proven to work very well in practice because it reliably detects uncertain situation where even consecutive unique hypotheses have large discrepancies. It robustly filters outliers, e.g. if pRANSAM fails to compute the correct robot pose which normally is a rare event. As a summary, the algorithm may terminate if

$$\overline{r\left(\boldsymbol{C}_i\right)} > \delta_r \wedge d^2\left(\boldsymbol{r}_h^e, \boldsymbol{R}_h^e\right) < {\xi_r}^2 \wedge \sigma_{med} < \xi_{med}. \tag{5.17}$$

However, in standard (office) environments $\overline{r\left(\boldsymbol{C}_i\right)} > \delta_r$ is often sufficient for excellent results.

5.3.3. Adaption of the Matching Threshold

In Sec. 5.1.1, only ideal situations were considered. More precisely, the environment of the robot is assumed to be static, i.e. the map represents the real workspace correctly and no human beings or other mobile objects share the workspace with the robot. Note that this is not the case in reality. E.g., furniture like tables or chairs are often moved to different places and after some time, the discrepancy of some parts of the map and the real workspace is significant. As a consequence, a fixed Ω_{thres} entails a considerable loss of hypotheses, i.e. the matching algorithm will not be able to provide hypotheses if the local environment exhibits too many changes. Thus, the threshold Ω_{thres} should be adapted permanently to the current situation. Therefore, Ω_{thres} is computed as

$$\Omega_{thres}(T) = \frac{1}{\tau} \sum_{t=T-\tau_{tr}}^{T} \frac{1}{\mid \mathcal{H}(t) \mid} \sum_{j=1}^{|\mathcal{H}(t)|} \Omega\left(\boldsymbol{H}_j\right)(t). \tag{5.18}$$

The robot has performed T localization steps and the current Ω_{thres} is calculated as the average of the average quality of all hypotheses computed in one localization step. In equation (5.18) $\mathcal{H}(t)$ is the set of all hypotheses of the t-th localization step and $\Omega\left(\boldsymbol{H}_j\right)(t)$ is the matching quality of the j-th hypotheses of localization step t. The parameter τ_{tr} determining the size of the sliding window needs to be chosen carefully. If τ_{tr} is small, the new matching threshold is very sensitive to environmental changes. Conversely, if τ is too large, Ω_{thres} adapts only slowly to the current situation.

However, a significant drawback of this threshold adaption is that is does not take the variance of the clusters' quality into account and thus is often far too optimistic. Ω_{thres} tends to increase permanently. Consequently, the map matching algorithm is very likely to provide not even a single hypothesis if Ω_{thres} is too large. This phenomenon was already reported in our previous work [21]. In this case, a zero quality was simply added to the queue and Ω_{max} decreased until pRANSAM computes new hypotheses again.

Unfortunately, this is a very undesirable behavior because then it makes the decision whether a unique robot pose can be determined or not very hard. In order to cope with this problem, Bollinger Bands [22] are employed as an indicator for a suitable threshold update. Bollinger Bands are a technical tool for stock chart analysis and consist of three curves. The so-called middle band mb_N usually is a simple moving average of the last n days. The upper band ub_N and the lower band lb_N are calculated by adding and subtracting the standard deviation k times. More formally, the three curves are defined by

$$\begin{aligned} mb_N &= \overline{C}_N = \frac{1}{n} \sum_{i=N-n}^{N} C_i \\ ub_N &= \overline{C}_N + k\sigma \\ lb_N &= \overline{C}_N - k\sigma \end{aligned}$$

One example is given in Fig. 5.4. The price of a fictional stock chart is plotted against the days. In this example, $N = 10$ and $k = n = 3$. The red squares represent the price of the stock. The black curve is the moving average while the magenta and green curve depicts the upper and lower band, respectively.

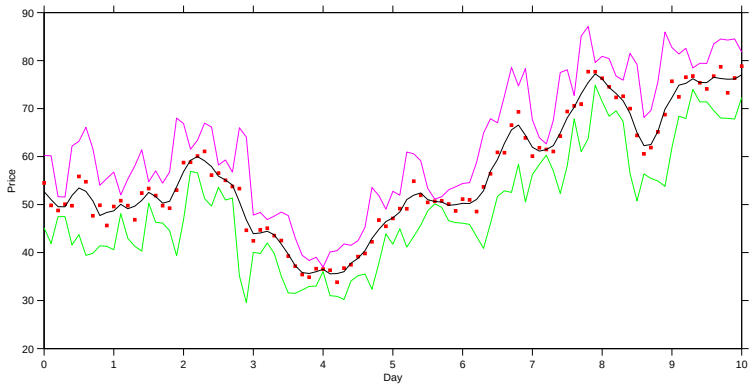


Figure 5.4.: The red squares indicate the stock price. The black curve shows the moving average. The magenta curve depicts the upper band while the green curve represents the lower band.

In order to transfer the algorithm to the localization framework, the standard deviation of the hypotheses quality is computed as

$$\sigma_\Omega(T) = \sqrt{\frac{1}{\tau_{tr}} \sum_{t=T-\tau_{tr}}^{T} \left(\Omega_{thres}(t) - E\left(\Omega_{thres}\right)(T)\right)^2} \tag{5.19}$$

with

$$E\left(\Omega_{thres}\right)(T) = \frac{1}{\tau_{tr}} \sum_{t=T-\tau_{tr}}^{T} \Omega_{thres}(t). \tag{5.20}$$

Due to the characteristics of the hypotheses generation problem, the lower band is of interest. Thus, the new threshold is set to

$$\Omega'_{thres}(T) = \Omega_{thres}(T) - k\sigma_\Omega(T). \tag{5.21}$$

Moreover, for practical reasons, it is demanded that $\delta_l < k\sigma_\Omega(T) < \Delta_u \Omega_{thres}(T)$ with $\delta_l, \Delta_u \in [0; 1]$. This is a feasible restriction since it allows for the avoidance of significant deviations from the average and $k\sigma_\Omega(T) < \Omega_{thres}(T)$ can be guaranteed. Additionally, $\Omega'_{thres}(T)$ cannot be arbitrarily close to the average which would yield the same problems as described above.

Of course, the application of Bollinger Bands does not guarantee that pRANSAM always computes at least one hypotheses H with $\Omega(H) \geq \Omega_{thres}$ but it significantly mitigates the effect of the complete absence of such hypotheses. For a faster localization, the rule 5.17 is *softened* to

$$\exists \boldsymbol{C}_i(t) \forall t' > t : \overline{r(\boldsymbol{C}_i)}(t) > \delta_r \rightarrow \left(\overline{r(\boldsymbol{C}_i)}(t') > \delta_r \vee d^2(\boldsymbol{r}_h^e, \boldsymbol{R}_h^e)(t') < {\xi_r}^2 \right) \wedge \sigma_{med}(t') < \xi_{med}. \tag{5.22}$$

as soon as $\overline{r(\boldsymbol{C}_i)} > \delta_r$ is satisfied for the first time. This results in the fact that the best hypothesis H_{best} generated by pRANSAM can be employed for localization and H_{best} is used to compute $\boldsymbol{x}_h$. This does not necessarily mean that $\Omega(H_{best}) \geq \Omega_{thres}$. However, when 5.22 is violated, the hard localization criterion 5.17 applies again.

5.4. Local Map Consistency

As described in Sec. 5.1.1, the RBSM method introduced in Chap. 2 is used for generating the local map. For environments of small or medium size, RBSM provides excellent maps on average even without any sophisticated loop closing mechanism, as experiments have shown. Nevertheless, a drawback of the proposed localization framework is that the local map may diverge, i.e. the scan matching errors accumulate significantly over time. Although this problem is only of theoretical nature for standard office environments as considered in Chap. 2, more challenging workspaces like the MIT Killian Court (cf. Sec. 5.7 and Chap. 3.5) may entail a loss of consistency of the local map.

One option to elude this problem is to limit the size of the environment. The advantages are threefold. First, the probability of an inconsistent local map decreases the more its size is limited. Second, the computation time is nearly constant which is important for real-time applications. Third, the algorithm forgets areas which do not give any new information about the true robot pose or even distracts the system if the robot is moving in unknown regions (cf. Sec. 5.6). A clear disadvantage of this solution is that localization may take a longer time because the relative positions of several geometrical structures often determine the robot pose uniquely. Consider the example depicted in Fig. 5.1. Limiting the size of the local map would postpone the isolation of the correct pose until the robot reaches the upper right part of the map. However, the restriction of the map size is a simple mean to counteract the divergence problem which has proven to be very efficient in practice (cf. Sec. 5.7). As in Chap. 3, the map is subdivided into a set of consecutive fragments $\mathcal{F} = \{f_i \mid i \in [1, N]; N \in \mathbb{N}\}$ where N is the total number of fragments and each fragment $f_i = \{(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, ..., \boldsymbol{x}_{i_n}) | n \in \mathbb{N}\}$ consists of a subset of local map points. The idea is to match only the last m fragments $\mathcal{F}_m = \{f_i \mid i \in [N-m, N]; N, m \in \mathbb{N}, m < N\}$.

A second but more complicated option is to correct the local map by means of gradient descent. The idea is based on the assumption that the geometrical structure of

the environment around all hypotheses computed by pRANSAM is sufficient similar which facilitates to correct the relative transformation of the fragments. For example, the hypotheses of Fig. 5.1b do not allow for determining a unique robot pose but the environmental structure is similar enough to adapt the local map no matter which hypothesis is chosen. Thus, the global map is exploited in order to correct the local one. Note that Fig. 5.1 depicts an ideal artificial indoor environment. In real environments not all hypotheses are of the same quality. Consequently, the best hypotheses (respectively cluster) is chosen to correct the current fragment.

In order to explain the algorithm, some definitions are introduced first. Each fragment $f_i \in \mathcal{F}$ is given with respect to the origin $\mathcal{LM}$ of the local map. The correction step is applied to the current fragment f_N because all previous fragments have already been adapted. Hence, let ${}^{\mathcal{LM}}\boldsymbol{T}_{f_N}$ be the transformation from $\mathcal{LM}$ to f_N. Furthermore, each point $\boldsymbol{x}_{N_j} \in f_N$ is given with respect to $\mathcal{LM}$. The gradient descent manipulates the fragment f_N with respect to its origin. Consequently, the fragment points are expressed with respect to the base of f_N:

$$\boldsymbol{x}_{N_j}^{f_N} = \left({}^{\mathcal{LM}}\boldsymbol{T}_{f_N}\right)^{-1} \boldsymbol{x}_{N_j} \tag{5.23}$$

This point is corrected by applying a transformation $\boldsymbol{T}_{\delta x,\delta y,\delta\phi}$ where $\delta x, \delta y$ are translational parameters and $\delta\phi$ is the rotational component. Afterwards, this point is back-transformed to $\mathcal{LM}$ yielding a point $\boldsymbol{x}'_{N_j}$:

$$\boldsymbol{x}'_{N_j} = {}^{\mathcal{LM}}\boldsymbol{T}_{f_N}\boldsymbol{T}_{\delta x,\delta y,\delta\phi}\boldsymbol{x}_{N_j}^{f_N} \tag{5.24}$$

Let $\boldsymbol{H}_{best}$ be the current best hypothesis which transforms $\boldsymbol{x}'_{N_j}$ into the reference frame $\mathcal{GM}$ of the global map:

$$\boldsymbol{x}_{N_j}^{\mathcal{GM}} = \boldsymbol{H}_{best}\boldsymbol{x}'_{N_j} \tag{5.25}$$

Moreover, let $\boldsymbol{c}_{N_j}$ be the corresponding point of the global map which is located closest to $\boldsymbol{x}_{N_j}^{\mathcal{GM}}$. Now the objective is to minimize the following mean square error function:

$$e\left(\delta x, \delta y, \delta\phi\right) = \sum_{i=N_1}^{N_n} \left[\boldsymbol{x}_i^{\mathcal{GM}}\left(\delta x, \delta y, \delta\phi\right) - \boldsymbol{x}_i^{\mathcal{GM}}\right]^2 \tag{5.26}$$

To this end, a standard gradient descent method is applied which is formally described as

$$\boldsymbol{\delta}^{(j+1)} = \boldsymbol{\delta}^{(j)} - \alpha^{(j)}\nabla e\left(\boldsymbol{\delta}^{(j)}\right) \tag{5.27}$$

where α is the step size and $\nabla = \left(\frac{\partial}{\partial \delta x}, \frac{\partial}{\partial \delta y}, \frac{\partial}{\partial \delta \phi}\right)$ denotes the Nabla operator. A detailed derivation of ∇e can be found in appendix D;

5.5. Run Time Aspects

5.5.1. Reduction of the local Map

Hitherto at each iteration, the complete local map is matched to the global one. Consequently, the computation time increases linearly with the number of points which belong to the local map data. This aspect is negligible in small or even medium size environments. In particular, if the robot quickly enters very information rich parts of its workspace, the required travel distance until a unique robot pose can be computed is short. Hence, the size of the local map is moderate. In contrast to this considerations, the robot may be obliged to execute paths of substantial length if the size of its workspace exceeds standard office environments. Moreover, if the computational resources are limited, it may be unfeasible to match the complete data collected so far. Even if the size of the local map is limited as proposed in Sec. 5.4, matching all data points might require significant computation time.

One example for a localization path where data reduction is prudential is depicted in Fig. 5.5a. Due to structural ambiguity, the robot must explore almost the complete workspace until localization becomes possible. However, most of the data are useless because bare walls do not give any new information about the robot pose. Consequently, a large amount of local map information can be discarded. Obviously, the corners and parts of the corridor relative to the corners uniquely determine the robot pose. This example has already been given in Fig. 5.1. The robot follows approximately the same path. We assume that the number of clusters computed by pRANSAM is a discrete function $f(t)$ of the time t. For this example, $f(t)$ is shown in Fig.5.5b. At the beginning, the number of clusters decreases linearly until the robot reaches the first lower left corner of the map. Suddenly, the number of likely poses collapses to three as marked by the light red circles of Fig. 5.5a. The first and second derivative of $f(t)$ are shown in Fig. 5.5c and 5.5d. At $t = 5$ the first derivative has a local minimum which is highlighted by the red circle of Fig. 5.5c. Furthermore, the second derivative exhibits a zero crossing between $t = 5$ and $t = 6$ which entails a change of sign from minus to plus. The standard formulas for computing the discrete derivatives are applied:

$$\begin{aligned} \frac{\partial f(t)}{\partial t} &= f(t) - f(t-1) \\ \frac{\partial f^2(t)}{\partial t^2} &= f(t) - 2f(t-1) + f(t-2) \end{aligned} \tag{5.28}$$

At $t = 9$ the number of poses changes from three to two as marked by the yellow circles of Fig. 5.5a. A cluster in the upper right part of the map can be more or less

excluded because a considerable part of the local map would not have a corresponding point and thus the matching quality would be comparatively poor. However, the first derivative shows a local minimum again (yellow circle) and also the second derivative exhibits a zero-crossing between $t = 9$ and $t = 10$. The robot pose gets unique at $t = 12$ which is highlighted by the green circle. Both the first and second derivative exhibit the same characteristic as before. Accordingly the set of observations taken at $t = 0$,$t = 5$, $t = 9$, and $t = 12$ suffices to determine the robot pose. The first observation is necessary in order to treat a pose hypothesis in the lower left part of the map as an outlier (cf. Fig. 5.2a). The difference of the complete local map and its reduced version is shown in Fig. 5.5e and 5.5f. The ratio of their cardinalities is remarkable. Hence, the analysis of the second derivative of $f(t)$ is a feasible heuristic to considerably reduce the amount of data to be matched and thus to reduce the computational burden. Formally, a measurement is accepted for matching if

$$\frac{\partial f^2(t)}{\partial (t-1)^2} < 0 \wedge \frac{\partial f^2(t)}{\partial t^2} > 0 \wedge \frac{\partial f^2(t)}{\partial t^2} + \frac{\partial f^2(t)}{\partial (t-1)^2} > \Delta(|\mathcal{C}(t)|). \tag{5.29}$$

Note that $\Delta : \mathbb{N} \rightarrow \mathbb{N}$ is a function of the number of clusters computed at time t. In practise, $f(t)$ does not yield a linear decrease as illustrated in Fig. 5.5b. In particular, if $|\mathcal{C}(t)|$ is large, the second derivative may exhibit zero crossings which are due to occasional suboptimal matchings. These observations do not necessarily contain the desired information level. In addition, empirical investigations showed that the difference calculated in Eqn. 5.29 is often larger at the beginning of the localization. Therefore, a piecewise linear function is defined which maps Δ to $[\Delta_{max}, 2]$ with $\Delta_{max} \in \mathbb{N}$:

$$\Delta(|\mathcal{C}(t)|) = \begin{cases} \Delta_{max} & \text{if } |\mathcal{C}(t)| \geq |\mathcal{C}|_{max} \\ \lfloor \frac{|\mathcal{C}(t)|}{|\mathcal{C}|_{max}} \Delta_{max} + 2 - \frac{2 \cdot |\mathcal{C}(t)|}{|\mathcal{C}|_{max}} \rfloor & \text{otherwise} \end{cases} \tag{5.30}$$

A second (obvious) option to avoid the noisy data problem would be to low-pass filter $f(t)$. The drawback is that localization might be unnecessarily stalled since filtering entails matching data gathered in the past.

Note that there exist other useful techniques for data reduction. For example, Meyer-Delius and Burgard [23] proposed a method for down-sampling the data of a given map. This technique interprets a sample-based map as a probabilistic mixture model. Each point of the subset corresponds to an element of the model. The sample-based map is initialized using a particular grid-based sampling algorithm. Then a gradient ascent is applied in order to maximize the likelihood of the model parameters. The results presented in their work are very promising, e.g. a map of the Intel Research Laboratory (cf. Fig. 2.11) was down-sampled to 0.02 percent of the original amount of data points. However, a gradient ascent normally is slow compared to the zero-crossing analysis proposed above. In the work of [23], computation time is not of significant importance because down-sampling a navigation map is an offline procedure while the

data reduction algorithm presented here is employed online and thus computational complexity is an important issue.

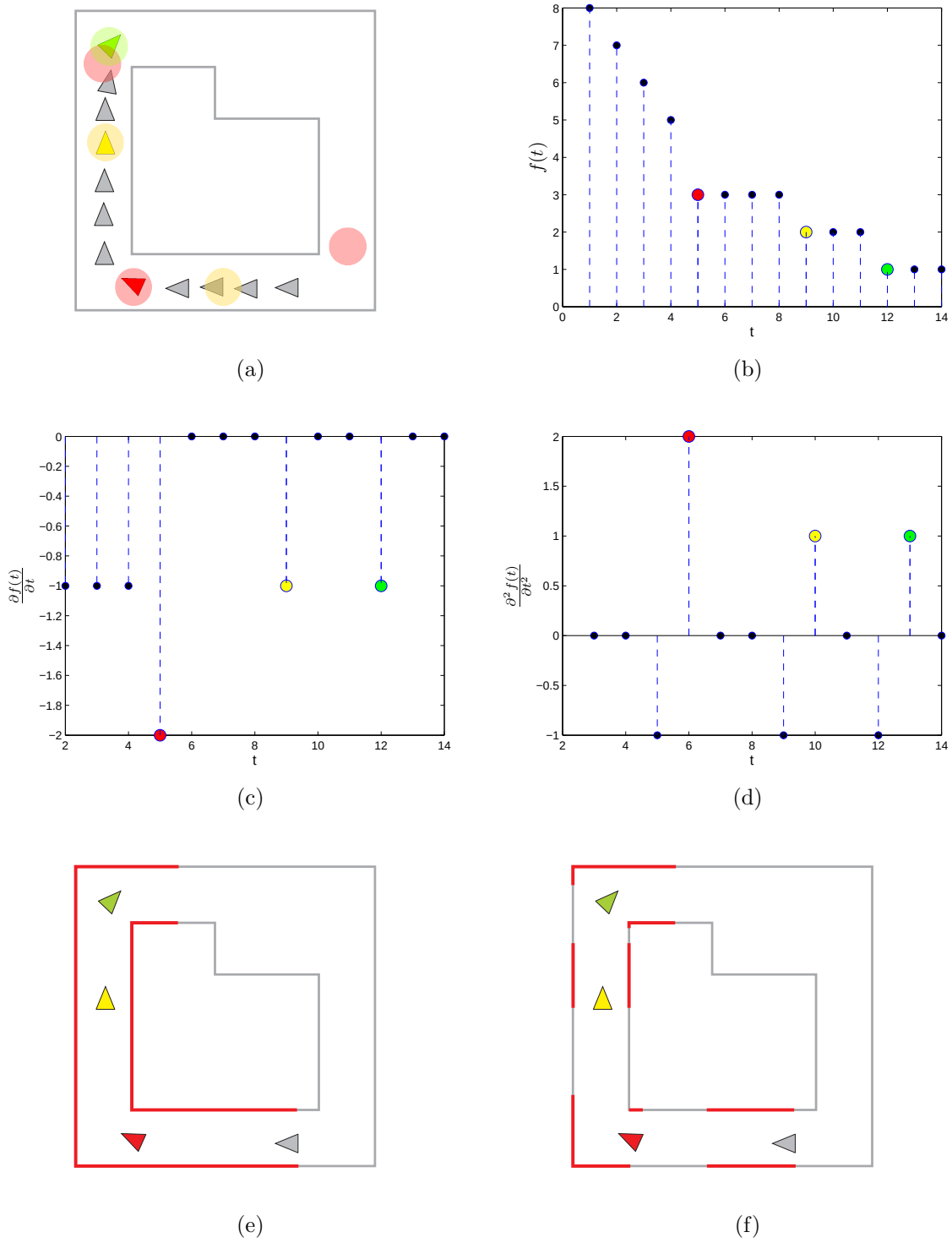


Figure 5.5.: a) The path of the robot and the information rich poses which facilitate the localization. b) The number of cluster as a function $f(t)$ of the time t. c) The first derivative $\frac{\partial f(t)}{\partial t}$ of $f(t)$. d) The second derivative $\frac{\partial f^2(t)}{\partial t^2}$ of $f(t)$. e) The complete local map after localization f) The local map after discarding all observation but the ones where $\frac{\partial f^2(t)}{\partial t^2}$ exhibits a zero-crossing.

5.5.2. Hash Table Precomputation

A further option of reducing the computation time during localization is to precompute the point triples of the global map relation table in an offline procedure. Intuitively, a pre-filled hash table considerably raises the probability of finding a good match because collisions occur from the beginning of the matching routine. Consequently, all T threads (compare Chap. 4.4.2) work on the same relation table since no more point triples need to be registered to the second relation table.

The first issue to examine is to calculate an estimation of the number of iterations necessary to pre-fill the relation table. To this end, without loss of generality, let $|B| \leq |A|$. Hence, relation table R_A is precomputed. Eqn. 4.19 provides the mean of the number point triples stored in R_A after h iterations. Ideally,

$$E'\{p_A(h)\} = \mathcal{M}_e \left(1 - \left(1 - \frac{1}{\mathcal{M}_e}\right)^h\right) = \mathcal{M}_e. \tag{5.31}$$

However, this is intractable because $h \to \infty$ in order to satisfy this equation. Therefore, constraint 5.31 is eased to

$$E'\{p_A(h)\} = \mathcal{M}_e \left(1 - \left(1 - \frac{1}{\mathcal{M}_e}\right)^h\right) = \mathcal{M}_e - \epsilon \tag{5.32}$$

with $\epsilon > 0 \wedge \epsilon \in \mathbb{N}$. Solving this equation for h yields

$$h = log_{1-\frac{1}{\mathcal{M}_e}} \left(\frac{\epsilon}{\mathcal{M}_e}\right) \tag{5.33}$$

Note that the smaller ϵ the larger h and thus a compromise must be found between the expected number of point triples in R_A and the number of necessary iterations. Given the expected number of point triples stored in R_A, the following theorem can be derived.
Theorem 2 (Precomputation of a Relation Table)**.** *Given $T \in \{k|k \in \mathbb{N} \wedge k \bmod 2 = 0\}$ threads and an appropriate ϵ, the success probability P_I' is equal or higher than P_I (cf. Theorem 1) if one relation table is pre-filled with $\mathcal{M}_e - \epsilon$ point triples and T threads draw from the same point set.*

Proof. Without loss of generality, let $|B| \leq |A|$. Thus, R_A is pre-filled and the expected number of point triples is $E\{p_A\} = \mathcal{M}_e - \epsilon$. As in Chap. 4.4.2, let $M_{tr} = \binom{|A|}{3}$ and $N_{tr} = \binom{|B|}{3}$. Then

$$\frac{\mathcal{M}_e - \epsilon}{N_{tr}} \geq \frac{E\{p_A(\tau i)\}}{N_{tr}} \tag{5.34}$$

and

$$\frac{\mathcal{M}_e - \epsilon}{N_{tr}} \geq \frac{E\{p_B((T-\tau)\, i)\}}{M_{tr}} \tag{5.35}$$

These are valid assumptions because for a suitable small ϵ the number of iteration i must be intractable large in order to violate the inequalities. Consequently,

$$\left(1 - \frac{\mathcal{M}_e - \epsilon}{N_{tr}}\right)^T \leq \left(1 - \frac{E\{p_B((T-\tau)\, i)\}}{M_{tr}}\right)^\tau \left(1 - \frac{E\{p_A(\tau i)\}}{N_{tr}}\right)^{T-\tau} \tag{5.36}$$

As a result $P_I' \geq P_I$. □

Note, that this theorem does not violate the characteristics derived in Chap. 4.4.2 because the preconditions have changed. Theorem 1 assumes both relation tables to be empty at the beginning and thus the inequalities 5.34 and 5.35 do not hold for this case. Moreover, ϵ must not necessarily be a small number. For example, if the relation tables are configured with 80 slots per axis, $\mathcal{M}_e \approx 512,000$. Empirical investigations showed that $h = 2,000,000$ yields good results. Hence, $\epsilon = 10,300$.

5.6. Detection of Unknown Regions

A last but important aspect of mobile robot localization is the detection of unknown places. This means that the robot could enter areas of its environment which are not registered in the global map and thus localization becomes impossible in such regions. For example, this case might occur if a door is suddenly opened during localization which was closed while the map data were collected allowing the exploration of new areas. One way to deal with this problem is to check how many points of the last τ_{up} observations are in contact with a point of the global map with respect to the best hypothesis found in the current localization step:

$$\begin{aligned} \mathcal{S} &= \left\{ \boldsymbol{S}_{T-\tau_{up}}, \boldsymbol{S}_{T-\tau_{up}+1}, ..., \boldsymbol{S}_T \right\} \\ \boldsymbol{H}_{best}(T) &= \max_{\Omega} \{ \boldsymbol{H} \mid \boldsymbol{H} \in \mathcal{H}(T) \} \\ \Omega'(T) &= \frac{1}{\mid \mathcal{S} \mid} \sum_{i=1}^{|\mathcal{S}|} \frac{1}{\mid \boldsymbol{S}_i \mid} \sum_{j=1}^{|\boldsymbol{S}_i|} contact_{Map}\left(\boldsymbol{y}_{j,i}\right) \\ \boldsymbol{y}_{j,i} &= \boldsymbol{H}_{best}(T) \cdot \boldsymbol{x}_{j,i} \end{aligned} \tag{5.37}$$

The set $\mathcal{S}$ represents the last τ' observations and $\boldsymbol{S}_k, k \in [T-\tau_{up}, T]$ is the set of points registered to the local map during the k-th observation. $\boldsymbol{x}_{j,i}$ is the j-th point of the local map of observation i which is transformed to the global map by the current best hypothesis $\boldsymbol{H}_{best}(T)$. The underlying idea of this approach implicates that only a few points of an unknown region are in contact with points of the global map because there

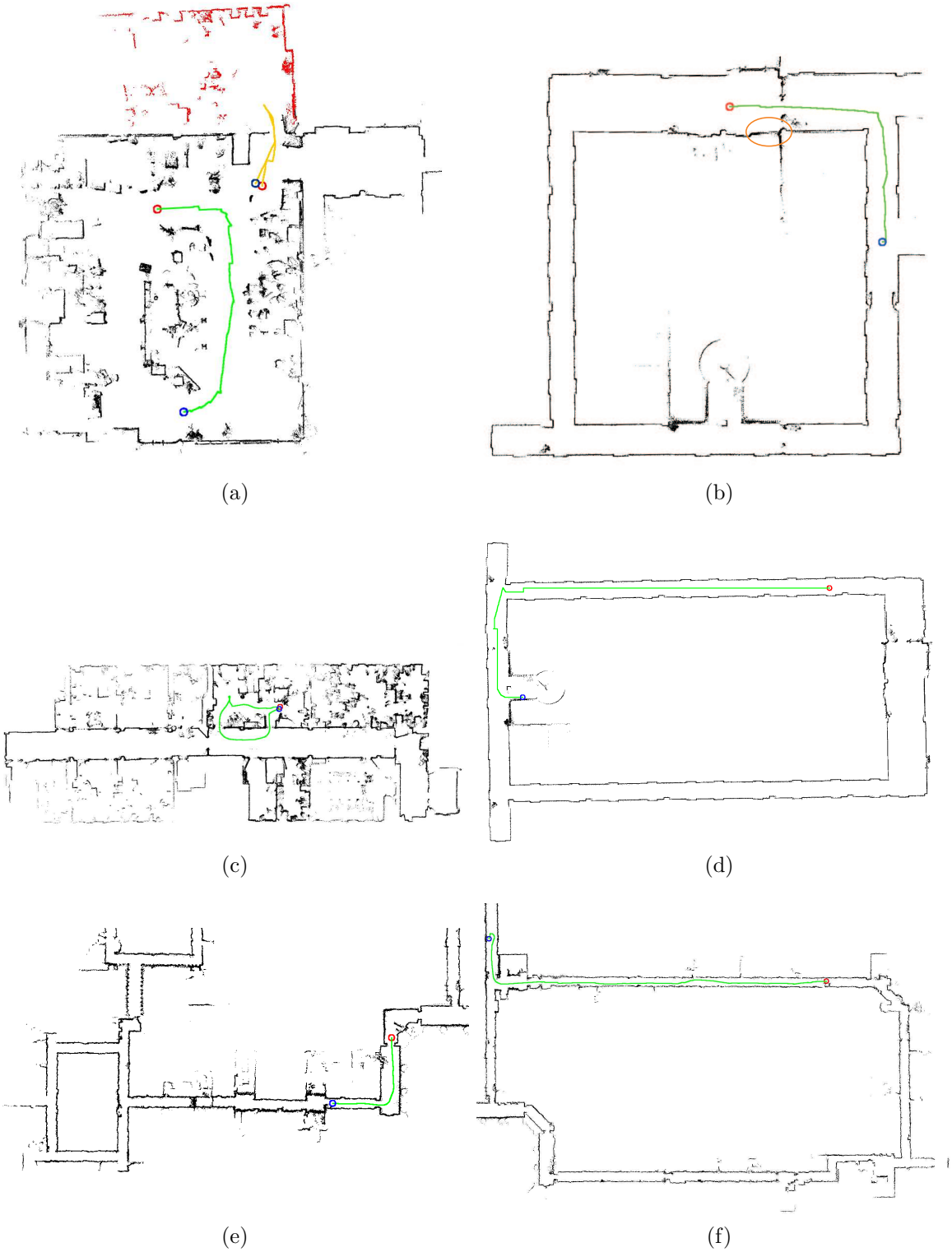
does not exist any point-to-point correspondence. Of course, this method may fail if the robot is located in a place which looks similar to another region registered in the global map.

5.7. Experimental Results

For evaluation of the proposed algorithm, a system equipped with an Intel Core 2 Duo E8300 'Wolfdale' processor running at $2.83GHz$, an ASUS P5Q-E motherboard, and 2 GB of RAM with a clock rate of $1066MHz$ was used. Moreover, a Windows XP OS with a Visual Studio 2005 compiler was employed. In the following section, experimental results are discussed based on four real world scenarios and one simulated data set. The first two real world scenarios represent the basement floor of the iRP robotics laboratory (cf. page 33) and its second floor respectively. The third and fourth data set is available at the Robotics Data Set Repository (Radish, cf. Chap. 2.4)[1]. The third environment is building 079 of the Autonomous Intelligent Systems (AIS) laboratory of the University of Freiburg (cf. Chap. 2.4). The fourth data set represents the well-known MIT Killian Court with its endless corridors. The simulated scenario looks quite similar to the second floor of the iRP robotics laboratory and is investigated due to its high structural ambiguity. The results presented in Sec. 5.7.1 and 5.7.2 may raise the impression of redundant information but the experimental evaluation was deliberately not restricted to less scenarios in order to demonstrate the robustness and applicability of the proposed algorithm to (almost) arbitrary environments.

At first, in Sec. 5.7.1 general characteristics of MML are discussed. To this end, several configurations as proposed in this chapter are compared. Considering the precomputation of one relation table and another setup where both relation tables are filled online makes the discrepancy of the computation time and the matching quality an interesting issue to examine. Details concerning the different configuration parameters are given in Sec. 5.7.1. In Sec. 5.7.2 MML is compared to Monte Carlo Localization (MCL) with respect to accuracy, computation time, and reliability. MCL was chosen as a benchmark algorithm since it is one of the most famous localization methods which exist today and it was employed in many challenging scenarios in the last decades. The accuracy of both MML and MCL is investigated by evaluating the rotational and translational distance to the ground truth. The latter refers to the Euclidean distance of estimated robot position and true position. This is a standard measure often used in mobile robotics to demonstrate the localization precision [24, 25, 26, 27, 28, 29, 30]. Additionally, the *success rate* is depicted for each scenario. A localization trial is defined as successful if the localization constraints are satisfied *and* the error drops below a certain threshold. The success rate was used in the literature to assess the effectiveness of relevant localization techniques [30, 26, 31, 27]. Typical error thresholds are 45 cm up to 1m. The general settings of pRANSAM and MML are given in the remainder of this section. The contact epsilon was set to $\varepsilon = 1.0$ (cf. Eqn. 4.7). The choice of ε strongly

[1] http://radish.sourceforge.net/



(a) (b)

(c) (d)

(e) (f)

Figure 5.6.: a) iRP robotics laboratory - The upper red part of the map highlights an unknown area. Two robot paths indicating different experiments are shown. b) First floor of the iRP laboratory - The red cicle marks a glass wall distracting the particles of MCL. c) fr079 d) Simulated scenario e) This path within the MIT Killian Court is used to compare MCL and MML. f) A second path within the MIT. It is employed to evaluate the characteristics of MML using several configurations.

depends on the geometrical structure of the point clouds to be matched. The closer the points are located to each other the smaller ε can be chosen. The quality threshold Ω_{thres} was initialized with 0.7. Furthermore, the size of the sliding window for updating the matching threshold needs to be defined (cf. Eqn. (5.18)). Empirical investigations showed that $\tau_{thres} = 10$ yields good results. Again, if τ_{thres} is too small, the algorithm is very sensitive to small environmental changes while τ_{thres} chosen too large results in a very slow adaption to the current situation. The queue size for the detection of unknown regions (cf. Eqn. (5.37)) was also set to τ_{up}=10. The maximum translational variance until splitting a cluster was set to $\sigma^2_{trans,max} = 30$ cm^2 and the maximal rotational variance was set to $\sigma^2_{rot,max} = 50^{\circ 2}$. A suitable choice for the parameters of the Bollinger Bands are $k = 4$, $\delta_l = 0.01$, and $\Delta_u = 0.07$ (cf. Eqn. (5.21)). Finally, the localization constraints must be configured. The cluster quality $\overline{r\,(\boldsymbol{C}_i)}$ must exceed $\delta_r = 0.8$. The squared Mahalanobis distance threshold was set to $\xi^2_r = 8$ and the uncertainty must not be larger than $\xi_{med} = 10.0$. Note, that these parameters were chosen once for all scenarios. No individual fine-tuning was necessary which would render the algorithm impractical.

The paths traveled by the robot are depicted in Fig. 5.6. A more detailed description is given in the according sections where also peculiarities are discussed.

5.7.1. Characteristics of MML

Four different MML configurations are compared in each scenario in order to discuss the advantages and drawbacks of the methods proposed in this chapter. For reader's convenience, the parameters are summarized in the following list and its abbreviations are introduced.

- Use Fixed Local Map Size - UFLMS: The first parameter determines whether the size of the local map is limited or not (cf. Sec. 5.4).
- Use Bollinger Bands - UBB: The second parameter enables or disables the usage of Bollinger Bands (cf. Sec. 5.3.3).
- Match All Local Map Points - MALMP: The third parameter specifies if all local map points are matched to the global map or only the observations carrying enough entropy (cf. Sec. 5.5.1).
- Use Local Map Correction - ULMC: The fourth parameter determines if the gradient descent as described in section 5.4 is applied.
- Precompute Relation Table - PRT: The pre-computation of the relation table is controlled by the fifth parameter (cf. Sec. 5.5.2).
- Matching Iterations - MI: The last parameter encapsulates the number of matching iterations (cf. Chap. 4.3).

The following tables list the different configurations which are regarded in the experiments.

Config	UFLMS	UBB	MALMP	ULMC	PRT	MI
1	×	✓	✓	×	×	100,000
2	×	✓	✓	×	✓	30,000
3	×	×	✓	×	×	100,000
4	×	✓	×	×	✓	30,000

Table 5.1.: Configuration Set 1

Config	UFLMS	UBB	MALMP	ULMC	PRT	MI
1	×	✓	✓	×	×	80,000
2	×	✓	✓	×	✓	30,000
3	×	✓	×	×	×	80,000
4	×	✓	×	×	✓	30,000

Table 5.2.: Configuration Set 2

Config	UFLMS	UBB	MALMP	ULMC	PRT	MI
1	✓	✓	✓	×	×	100,000
2	×	✓	✓	✓	×	100,000
3	✓	✓	✓	×	✓	30,000
4	✓	✓	✓	×	✓	60,000

Table 5.3.: Configuration Set 3

Config	UFLMS	UBB	MALMP	ULMC	PRT	MI
1	×	✓	✓	×	×	100,000
2	×	✓	✓	✓	×	100,000
3	✓	✓	✓	×	✓	30,000
4	✓	✓	×	×	✓	30,000

Table 5.4.: Configuration Set 4

Table 5.1 depicts the first configuration set applied in the two scenarios of the iRP robotics laboratory. It is employed in order to show the effect of pre-computing the relation table compared to entering point triples in both tables online. Furthermore, the limitation of local map points to information rich observations is discussed. The configuration set of table 5.2 is employed in the Freiburg scenario and compares local map reduction if the relation table is either precomputed or not. Predominantly, the third configuration set shown in table 5.3 keeps the size of the local map constant and compares precomputed and non-precomputed settings. It is applied in the MIT Killian Court experiment. After all, the configurations depicted in table 5.4 are used in the simulated environment in order to test the effect of the gradient descent for local map correction.

iRP Robotics Laboratory - Basement

The first experiment took place in the basement floor of the iRP robotics laboratory. The global map is depicted in Fig. 5.6a. The path of the robot is highlighted by the green curve. The initial pose is marked by the red circle while the final pose is marked by the blue circle. It has a size of 22 m $\times$ 14 m with 3 cm resolution. Note that the red area at the top does not belong to the global map. Moreover, this region is considered as an unknown area entered by the robot in a further experiment. The robot is equipped with a SICK LMS 200 laser range-finder. Fig. 5.7 and Fig. 5.8 depict the characteristics of MML in this scenario. The statistics are generated by averaging the results of 10 localization trials. The configuration settings shown in table 5.1 are applied. Fig. 5.7c shows that the robot is localized in a very early stage of the complete run. The configurations do not remarkably influence the localization success. However, a comparison of Fig. 5.7b and 5.7c highlight the effect of applying Bollinger Bands for the adaption of Ω_{thres}. Fig. 5.7b illustrates the percentage where $\overline{r\left(\boldsymbol{C}_i\right)} > \delta_r$. It is obvious that disabling Bollinger Bands significantly lowers the quality of the matching result (Configuration C3). The introduction of the soft localization constraint 5.22 facilitates acceptable success rates as proven by Fig. 5.7c. The number of clusters is plotted against the number of iterations in Fig. 5.7d. It clearly shows that the improved matching threshold adaption yields a much more reliable cluster generation. Disabling Bollinger Bands keeps the matching threshold high (cf. Fig. 5.8c) and the best matching result is on a comparable level (cf. Fig. 5.8d). Consequently, the probability of obtaining valid hypotheses is reduced. Furthermore, the gain of precomputing one relation table is astonishing in this scenario (curves C3 and C4). As can be derived from Fig. 5.7b, the probability of generating a unique cluster is even slightly higher while the computation time is very low (cf. Fig. 5.7a). In particular, configuration C4 saves computational resources since only 10 to 30 percent of the local map data are matched to the global map (cf. Fig. 5.8e). The accuracy of the localization is depicted in Figs. 5.7e and 5.7f. The ground truth was obtained by manually matching the laser data to the global map. Fig. 5.7e shows the translational distance to the ground truth while the rotational distance is depicted in 5.7f. Note the remarkable outliers in both figures. These outliers result from averaging the ground truth distance over *all* data which also include non-successful localization steps. The average translational distance over all successful localization steps ranges from 7.9 cm to 8.4 cm depending on the configuration. The average rotational distance is about 2°. Fig. 5.8b depicts the success rate which contains the percentage of matching results which are closer than 50 cm and 20° to the ground truth *and* the localization constraints are satisfied (cf. Eqn. 5.17 and 5.22). Opposite to that, the failure rate shown in Fig. 5.8a, represents the portion of localization steps where the localization constraints are satisfied and one of the aforementioned error bounds was exceeded. However, the failure rate is zero in this scenario.

iRP Robotics Laboratory - First Floor

The second experiment took place in the first floor of the iRP robotics laboratory. The global map is depicted in Fig. 5.6b. The path of the robot is highlighted again by the green curve. The initial pose is marked by the red circle while the final pose is marked by the blue circle. It has a size of 22 m $\times$ 24 m with the same resolution as in the first scenario (3 cm). Again, the robot is equipped with a SICK LMS 200 laser range-finder. Fig. 5.9 and Fig. 5.10 depict the characteristics of MML in this scenario. The statistics were generated by averaging the results of 10 localization trials. The configuration settings shown in table 5.1 were applied. The positive effect of Bollinger Bands is more distinctive in this experiment as proven by Fig. 5.9b and 5.9d because the probability of getting a unique cluster is much higher. In addition, cluster generation is much more likely. Nevertheless, Fig. 5.9c illustrates that the robot is localized with a 100 percent reliability independently from the configuration. The matching time is depicted in Fig. 5.9a. Again, precomputation significantly decreases the required computation time (curves C3 and C4). Discarding unnecessary local map data (cf. Fig. 5.10e) results in the highest matching threshold Ω_{thres} because all data points could be matched in almost every iteration on average (cf. Figs. 5.10c and 5.10d). The ground truth distance is depicted in Figs. 5.9e and 5.9f. The worst translational distance from ground truth is approximately 95 cm and the largest rotational error is 20°. Fig. 5.10b indicates that at some poses along the path it is impossible to localize the robot within the bounds as defined in the previous experiment. However, the average translational error is 8.6 cm when the system assumes a successful localization. The average rotational error is 1.6°. This implies that bad outliers are detected with a sufficient reliably. The error bounds were defined as 50 cm and 20°. The success rate and the failure rate are depicted in Figs. 5.10b and 5.10e. In summary, this scenario yields nearly the same characteristics as the first one viz. precomputing one relation table shows comparable accuracy at a compelling low computation time.

fr079

The third workspace is located at building 079 of the University of Freiburg. The global map and the associated robot path is depicted in Fig. 5.6c. It has a resolution of 3 cm and its dimensions are 37 m $\times$ 14 m. Fig. 5.11 and Fig. 5.12 depict the characteristics of MML in this scenario. The configuration settings shown in table 5.2 are applied. In contrast to configuration set 1 is that 80,000 iterations were chosen instead of 100,000 iterations if no precomputing is employed. Furthermore, local map reduction is enabled for both PRT = ✓ and PRT = ×. Roughly speaking, there are no peculiarities compared to the first two experiments. However, the average ground truth error for configuration C1 is conspicuous if no precomputation is enabled and the complete local map is matched to the global one (cf. Figs. 5.11e and 5.11f). If only successful localization trials are considered, i.e. the constraint 5.17 or 5.22 is satisfied, the average errors are 7 cm and 0.9° respectively which underlines the accuracy of the proposed approach. The success rate is depicted in Fig. 5.12b. Here, success means that the ground truth error may not

exceed 50cm and 20°, respectively. The failure rate shown in Fig. 5.12a is zero which emphasizes the reliability of the localization in this scenario. The validity of theorem 2 is confirmed again by Figs. 5.11b and 5.11c since the precomputation yields remarkable better results when matching the complete local map.

MIT Killian Court - Infinite Corridor

The MIT Killian Court is the most interesting scenario due to its exceptional dimensions. The complete global map is depicted in Fig. 3.11. The map resolution is 8 cm. The area has a size of approximately 250 m × 215 m. Fig. 5.6f shows a section of the map with a very long corridor traveled down by the robot. The path has a length of about 100 m and the robot had to cover a distance of about 85 m until a unique pose isolation occurred the first time. It is important to note that the laser data are very noisy and thus quite challenging since a lot of dynamic obstacles passed the field of view of the sensor during data acquisition. Table 5.3 depicts the according configuration settings. The statistics are shown in Figs. 5.13 and 5.14. As can be verified from Fig. 5.13a much computation time can be saved if precomputation is applied. Configuration C4 yields a higher computation time because C4 employs 60,000 iterations compared 30,000 iterations for C3. The reason for increasing the number of iterations were the suboptimal results for 30,000 iterations. The portion of successful localization steps per iteration are depicted in Fig. 5.13c. Robot localization gets realistic at iteration 280 where the curves of C1, C3, and C4 start to increase. The chance of robot localization employing configuration C2 is negligible because it does not limit the local map size. Furthermore, C2 applies local map correction which even amplifies the effect of local map inconsistency due to many dynamic obstacles crossing the sensor (cf. Fig. 5.14e). Consequently, the gradient descent fails to correct the local map. On the contrary, the probability that C2 generates a unique cluster with a sufficient rating is relatively high (cf. Fig. 5.13b). This is not a contradiction to Fig. 5.13c because the best matching quality Ω_{best} decreases permanently and thus the threshold Ω_{thres} is also kept on a low level (cf. Figs. 5.14c and 5.14e). The localization error is depicted in Figs. 5.13e and 5.13f. It collapses when the number of positive localization trials starts to increase at iteration 280. Considering only the positive localization results the average errors are (61.6 cm; 1,799 cm; 56,6 cm; 97,2 cm) and (3.58°; 46.8°; 4.04°; 4.1°) for (C1;C2;C3;C4). This implies that C2 does not yield an accurate pose estimation even if the localization constraints are satisfied. Note that a positive localization result does not necessarily mean that the result is accurate. The framework just believes that the pose is isolated with a sufficient precision. Hence, Fig. 5.14b depicts the probability that the localization error is not boundless. In this scenario a localization is considered as accurate enough when the translational error is smaller than 1 m and the rotational error does not exceed 20°. Fig. 5.14a clearly shows that a precise localization is very likely. In particular, configurations C3 and C4 exhibit comparable results. The failure rate is illustrated in Fig. 5.14b. The curve indicates that wrong localizations are a very rare event. Note the poor characteristics at the end where all localization results are erroneous for C4. Further experiments showed that this error is due to the translational distance. 80 Percent of the translational error are

smaller than 2 m. The fixed-size local map is depicted Fig.5.14f. Thus, precomputation is favorable since the discrepancy of computation time is remarkable as illustrated in Fig. 5.13a. To summarize the results, the proposed localization method proved to solve the global localization problem in an efficient manner even in large scale workspaces like the MIT Killian Court. Taking the size into consideration, it is very surprising that the small local map depicted in Fig. 5.14f suffices to determine a unique robot pose.

Simulated Environment

This scenario was generated in order to demonstrate the capability of coping with structural ambiguity and local map inconsistencies. Thus, localization is compared enabling and disenabling local map correction and a local map of fixed size. The global map and the tested robot path is depicted in Fig. 5.6d. It has a resolution of 6 cm and its size is 81 m $\times$ 47 m. A Sick LMS laser was simulated with a 180° field of view and a resolution of 0.5°. Gaussian noise was added to the robot motions with standard deviations of 4 cm and 4°, respectively. Furthermore, noise was added to the laser readings using a standard deviation of 5 mm. Configuration set 5.4 was used in this scenario. Configuration C2 employs the gradient descent while C1 does not. The statistics depicted in Figs. 5.15 and 5.16 show that the gradient descent indeed mitigates the effect of accumulated scan matching errors yielding inconsistent maps over time. First of all, the matching time shown in Fig. 5.15a indicates that the local map is more accurate after correction. Additionally, more well rated clusters are generated (cf. Fig. 5.15b) and a positive localization decision is made more often as illustrated in Fig. 5.15c. After matching the local map, more points are in contact with the global map if the gradient descent is applied (cf. Fig.5.15f) and thus Ω_{thres} (Fig. 5.15e) is higher. The error bounds for the success rate highlighted in Fig. 5.16f are 50 cm and 20°. Configurations C3 and C4 keep the local map size constant which yields better results because much information-poor data is discarded influencing the matching output. The failure rate is shown in Fig. 5.16e. The statistics indicate that the uncorrected local map yields much more outliers than the corrected one. It even doubles the maximal failure rate at iteration 188. Furthermore, configuration C4 yields a considerable higher portion of outliers than C3 which shows a drawback of our data reduction technique in this scenario. Fig. 5.16a and 5.16b depict the uncorrected and corrected map, respectively. It clearly shows that the corrected map is more accurate. Fig. 5.16c and 5.16d show local maps with bounded size. Data reduction is applied in Fig. 5.16d. Although only very few map points are left, the information is sufficient to localize the robot.

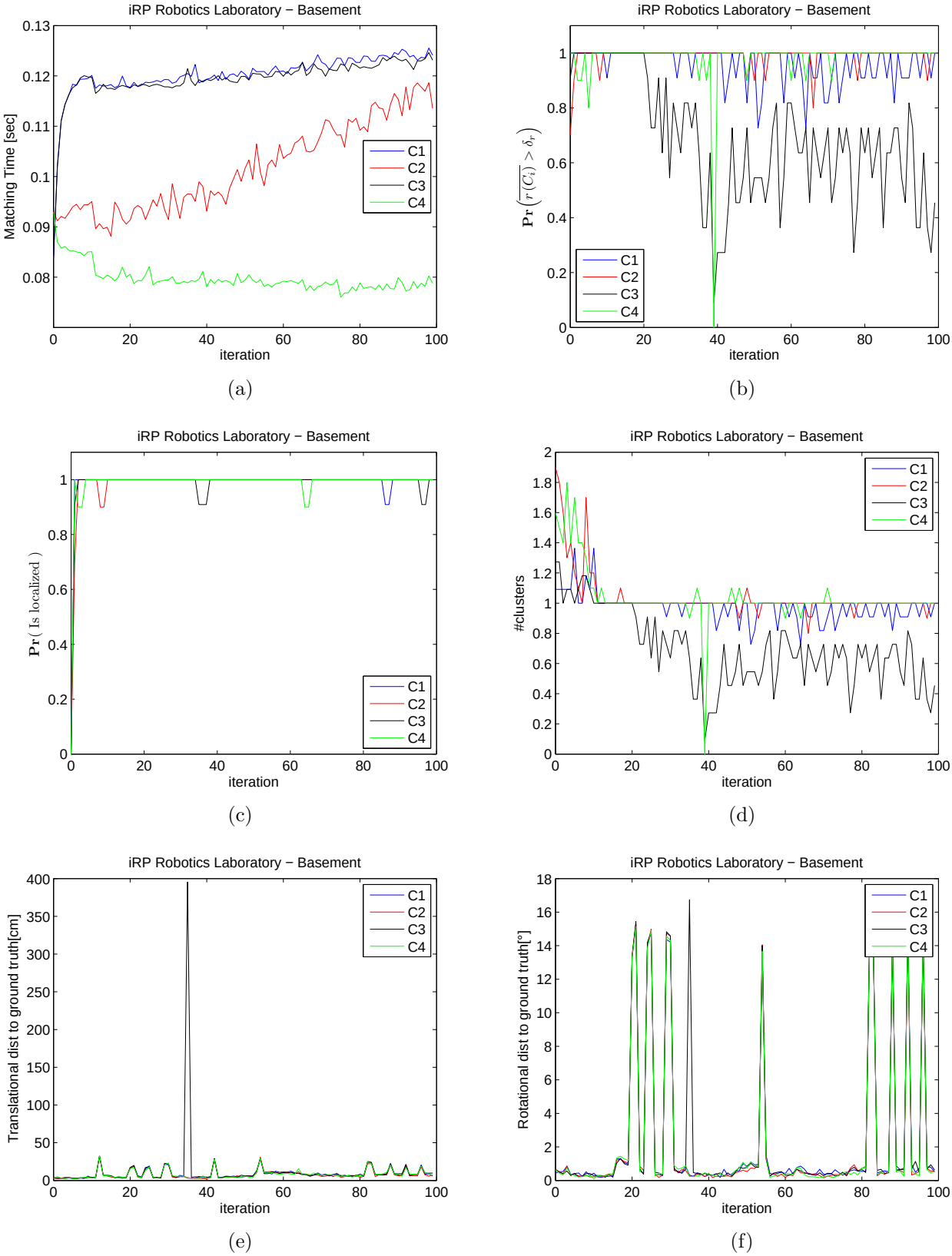
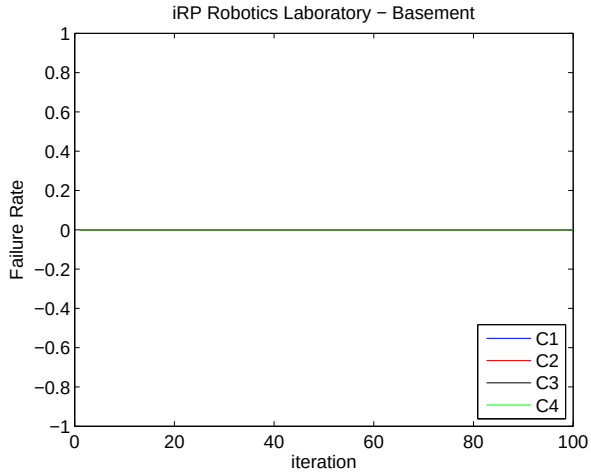
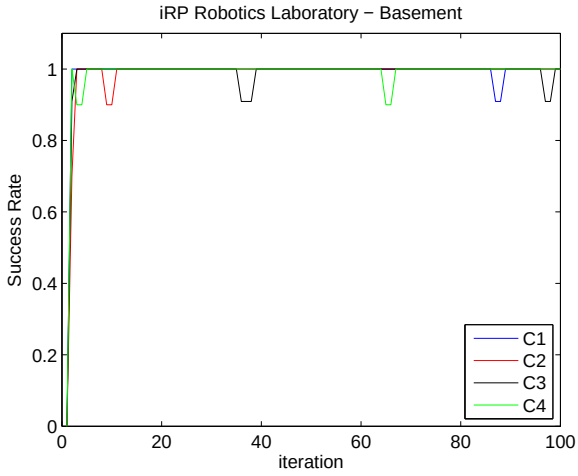


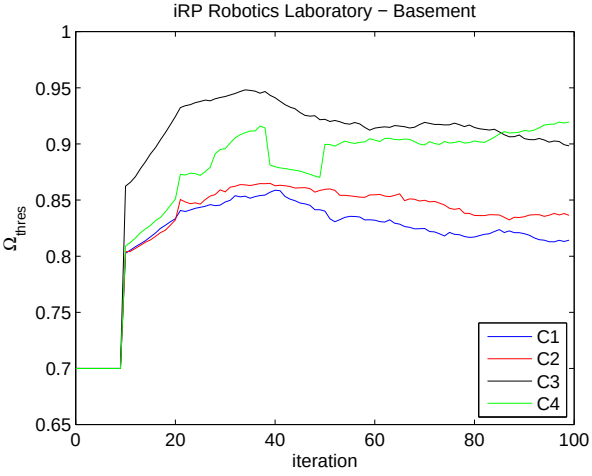
Figure 5.7.: iRP Basement - a) Matching time b) Probability that a good cluster has been generated c) Probability that the localization constraints are satisfied d) Number of clusters e) Translational distance to ground truth f) Rotational distance to ground truth



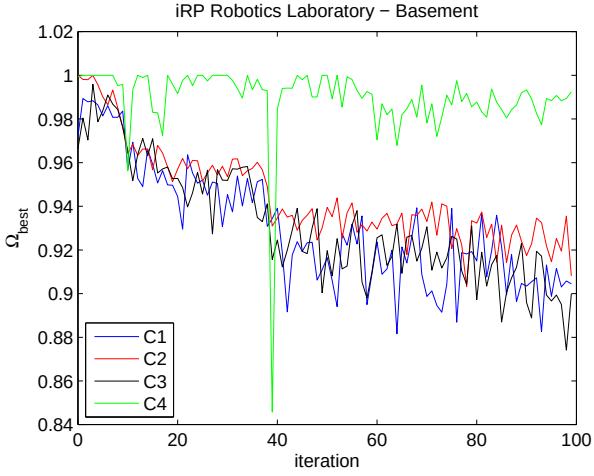
(a)



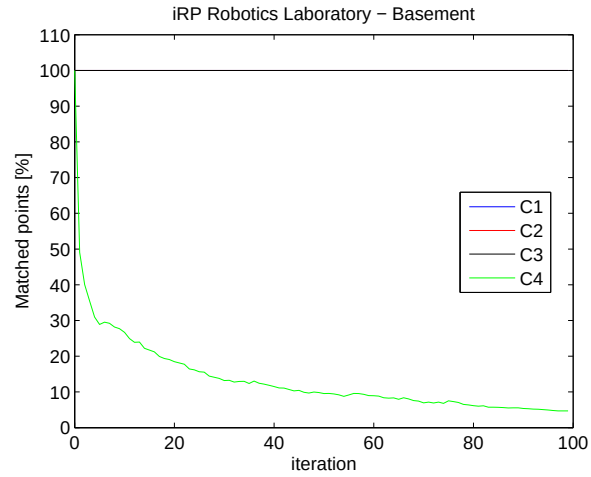
(b)



(c)



(d)



(e)

Figure 5.8.: iRP Basement - a) Failure rate b) Success rate c) The matching quality threshold d) The best matching quality e) Portion of the total number of points which is matched to the global map

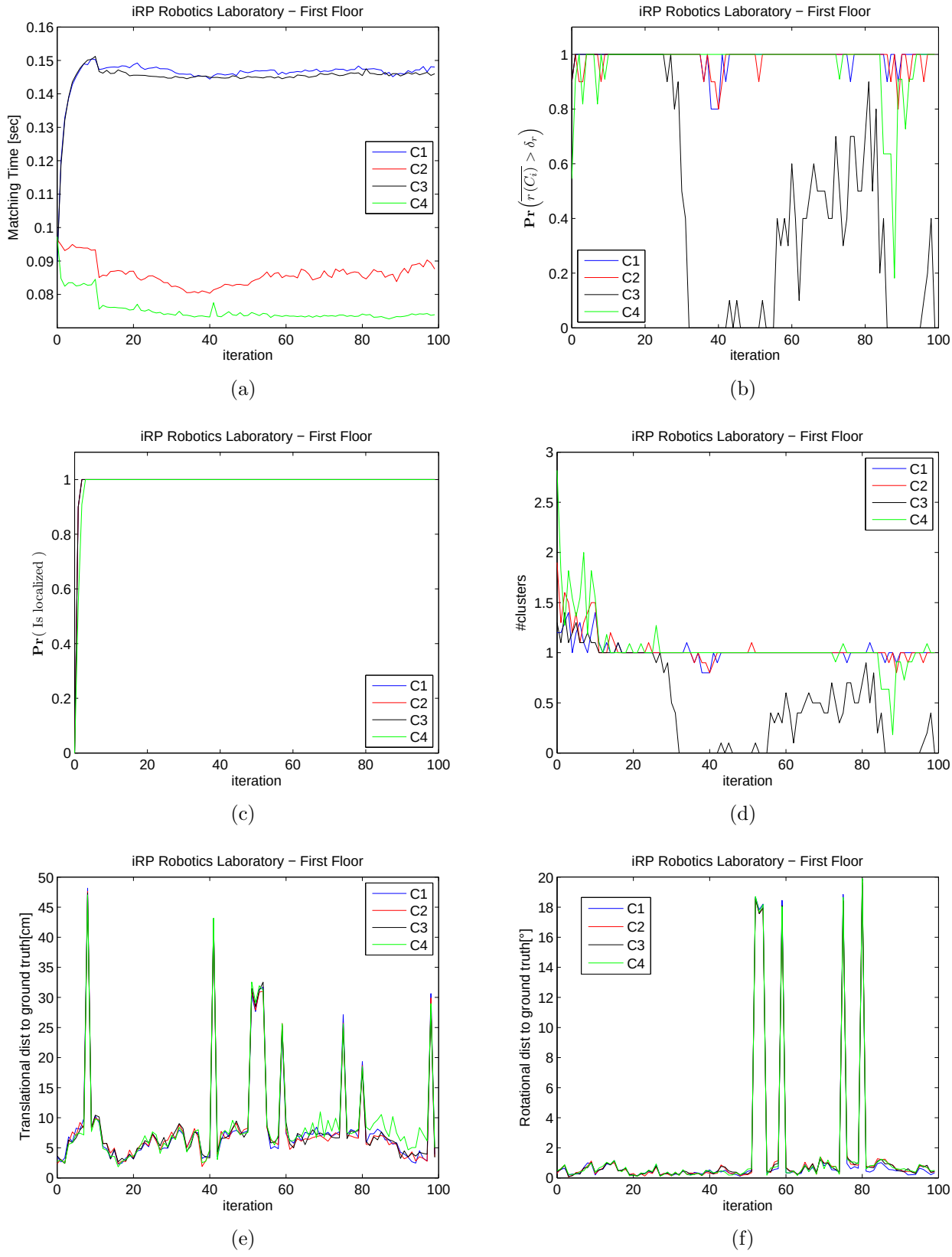
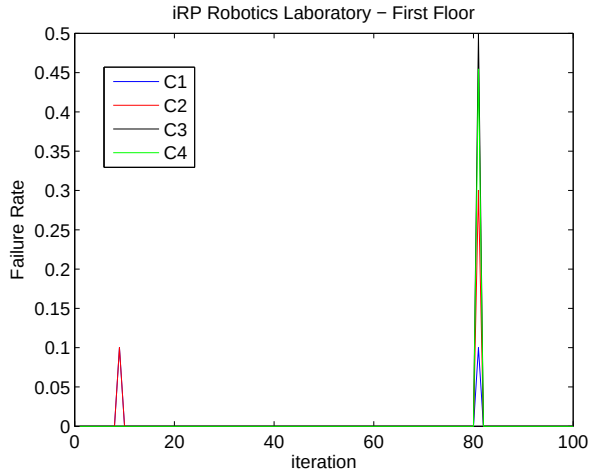
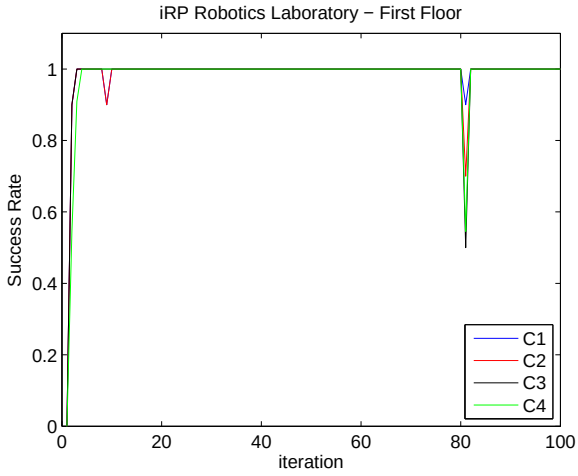


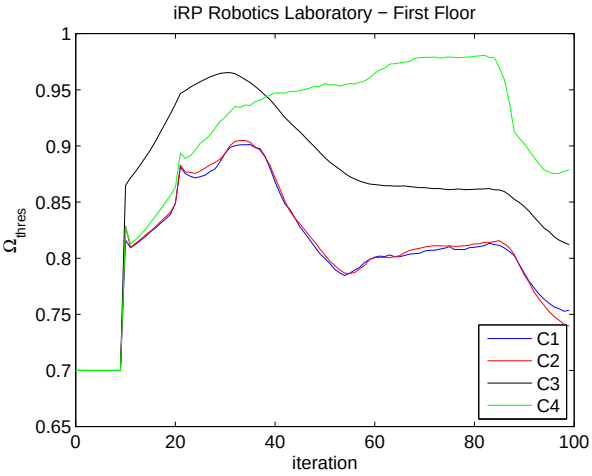
Figure 5.9.: First floor of the iRP robotics laboratory - a) Matching time b) Probability that a good cluster has been generated c) Probability that the localization constraints are satisfied d) Number of clusters e) Translational distance to ground truth f) Rotational distance to ground truth



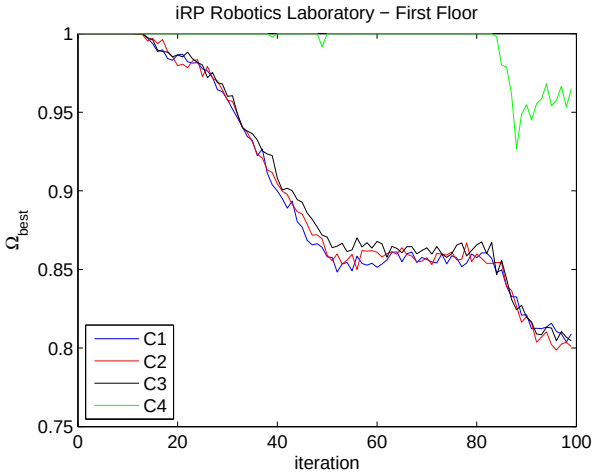
(a)



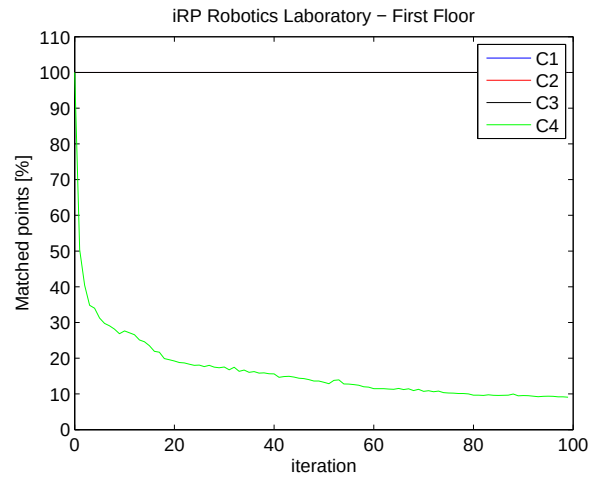
(b)



(c)



(d)



(e)

Figure 5.10.: First floor of the iRP robotics laboratory - a) Failure rate b) Success rate c) The matching quality threshold d) The best matching quality e) Portion of the total number of points which is matched to the global map

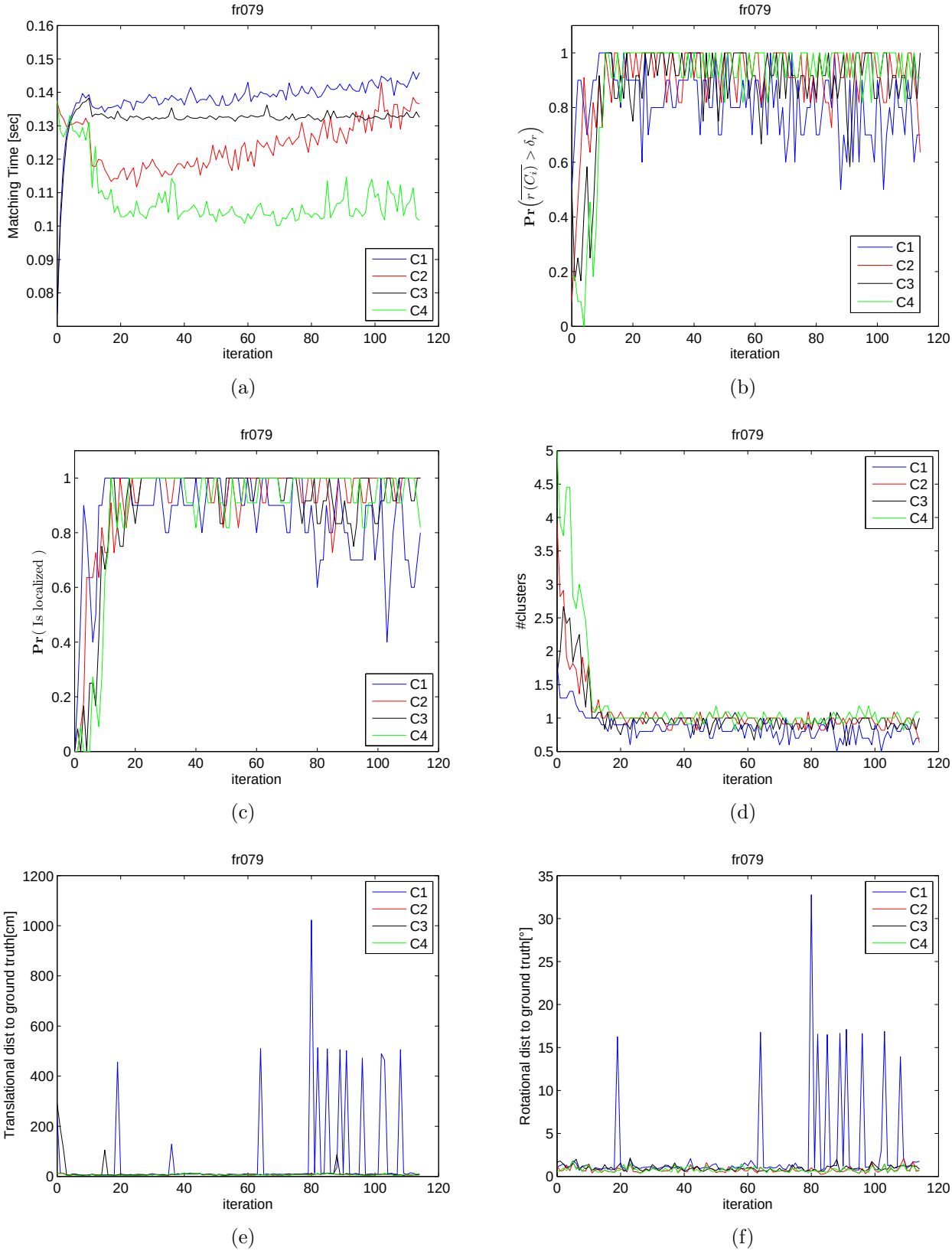
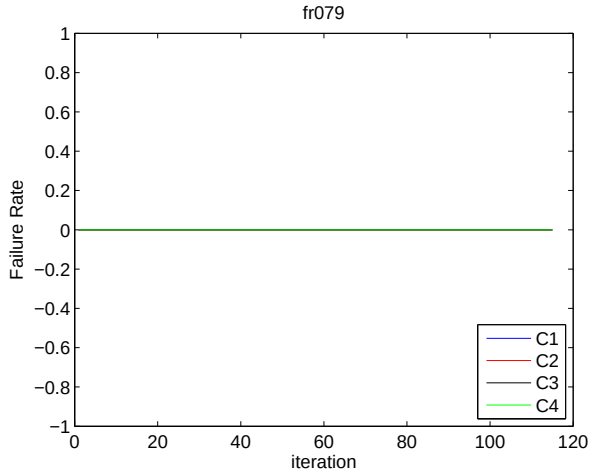
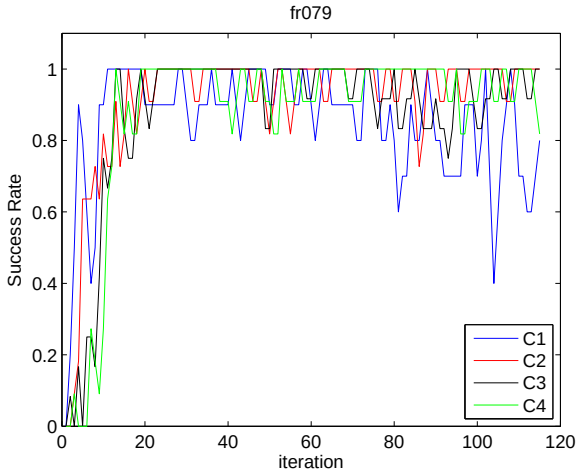


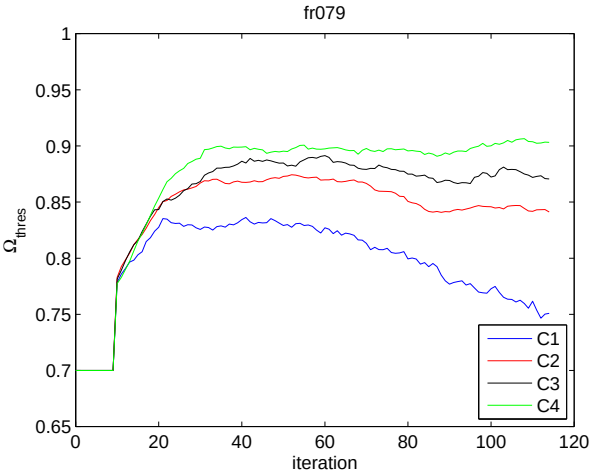
Figure 5.11.: Building 079 of the University of Freiburg - a) Matching time b) Probability that a good cluster has been generated c) Probability that the localization constraints are satisfied d) Number of clusters e) Translational distance to ground truth f) Rotational distance to ground truth



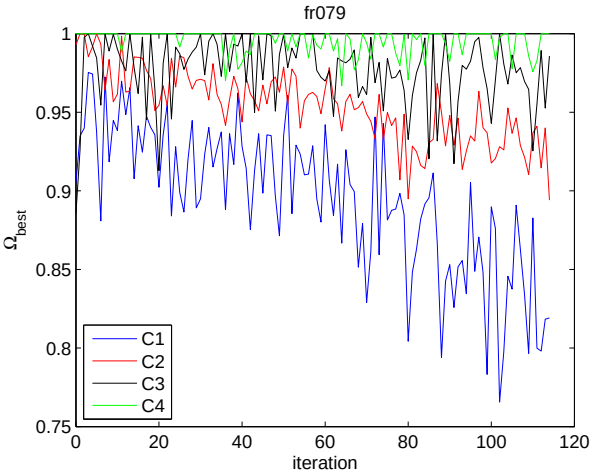
(a)



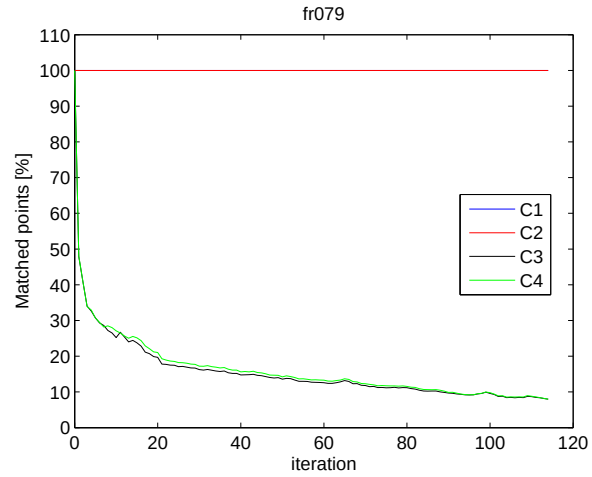
(b)



(c)



(d)



(e)

Figure 5.12.: Building 079 of the University of Freiburg - a) Failure rate b) Success rate c) The matching quality threshold d) The best matching quality e) Portion of the total number of points which is matched to the global map

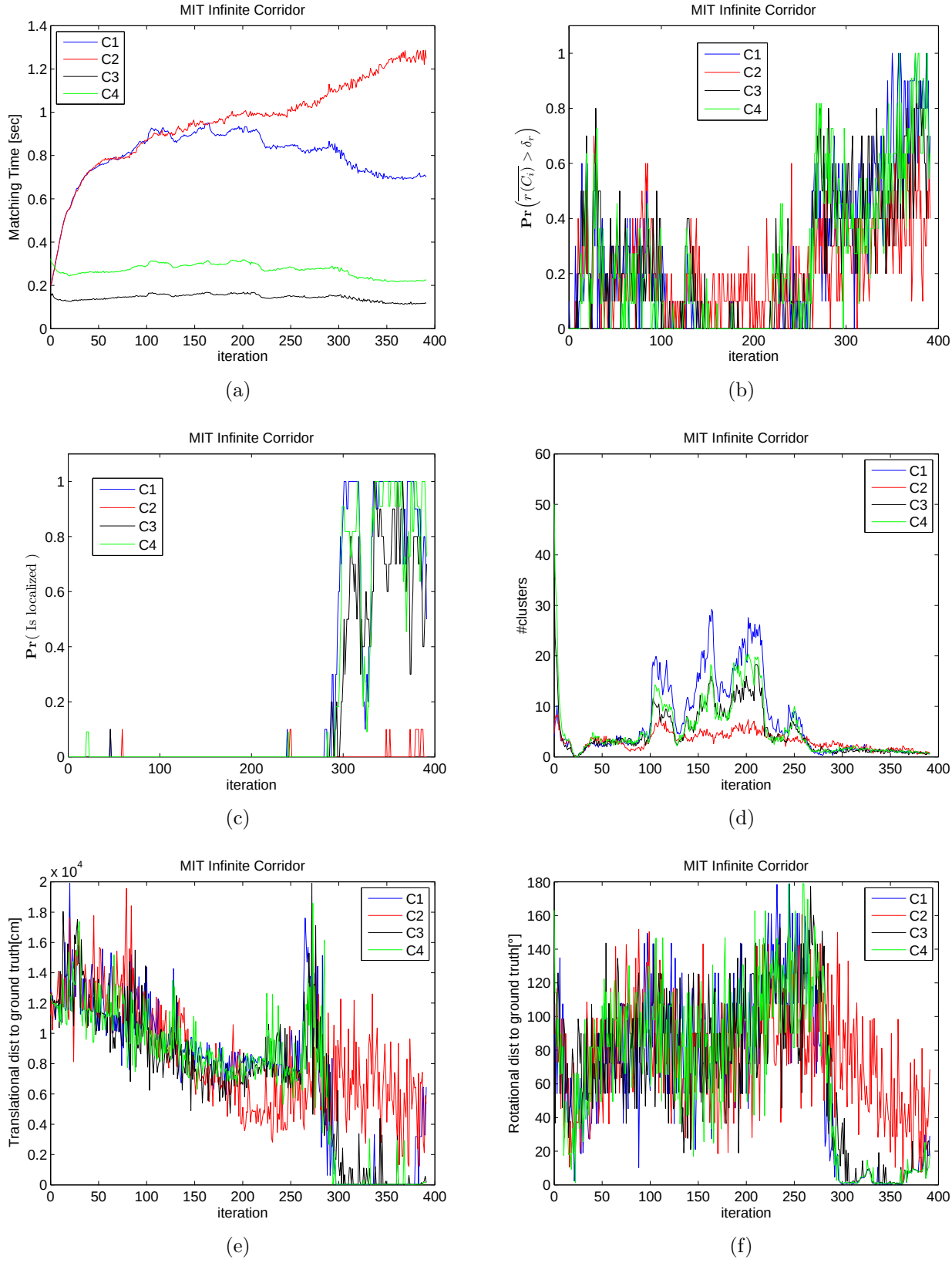
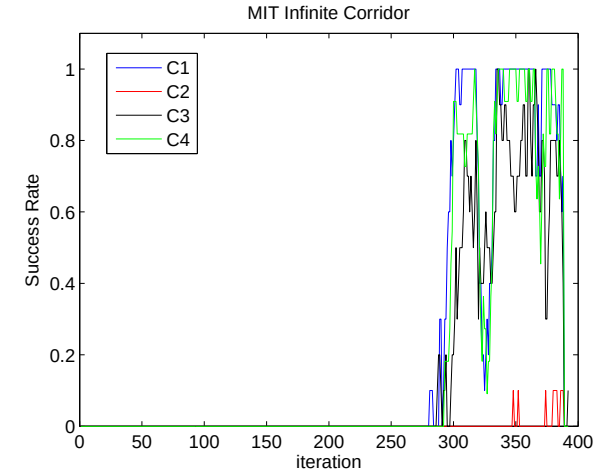
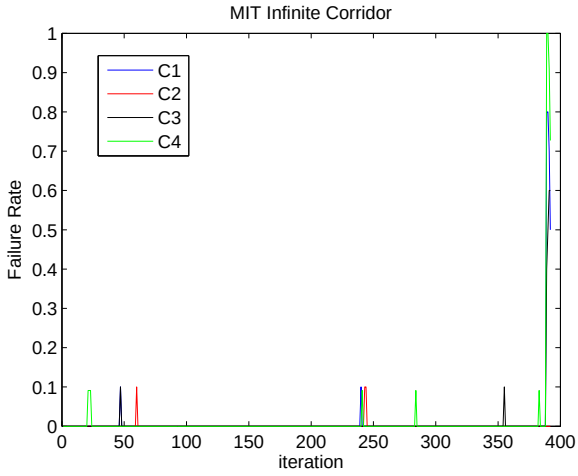


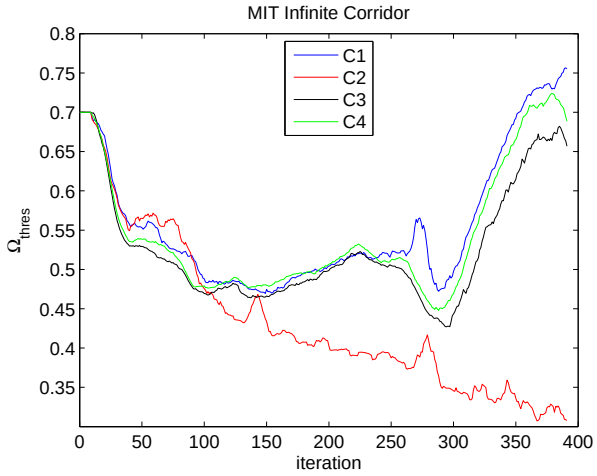
Figure 5.13.: MIT Killian Court infinite corridor - a) Matching time b) Probability that a good cluster has been generated c) Probability that the localization constraints are satisfied d) Number of clusters e) Translational distance to ground truth f) Rotational distance to ground truth



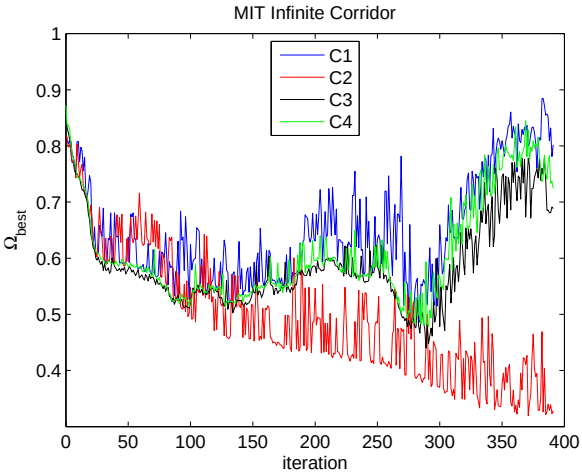
(a)



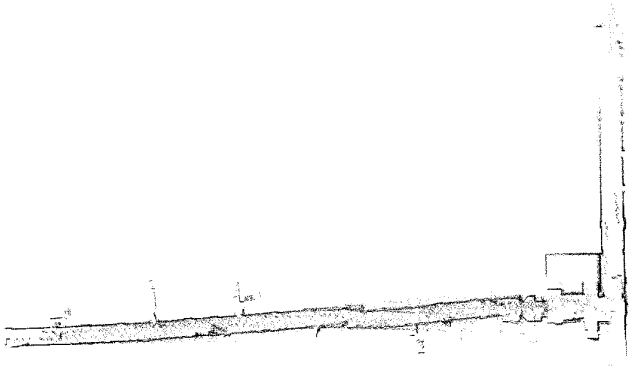
(b)



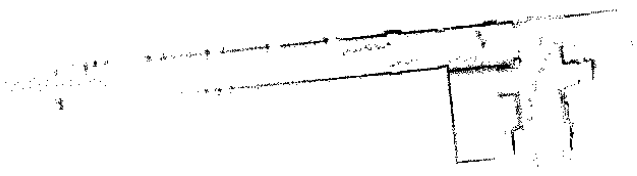
(c)



(d)



(e)



(f)

Figure 5.14.: MIT Killian Court infinite corridor - a) Success rate b) Failure rate c) The matching quality threshold d) The best matching quality e) The complete local map after application of the gradient descent approach for map correction f) The local map with limited size

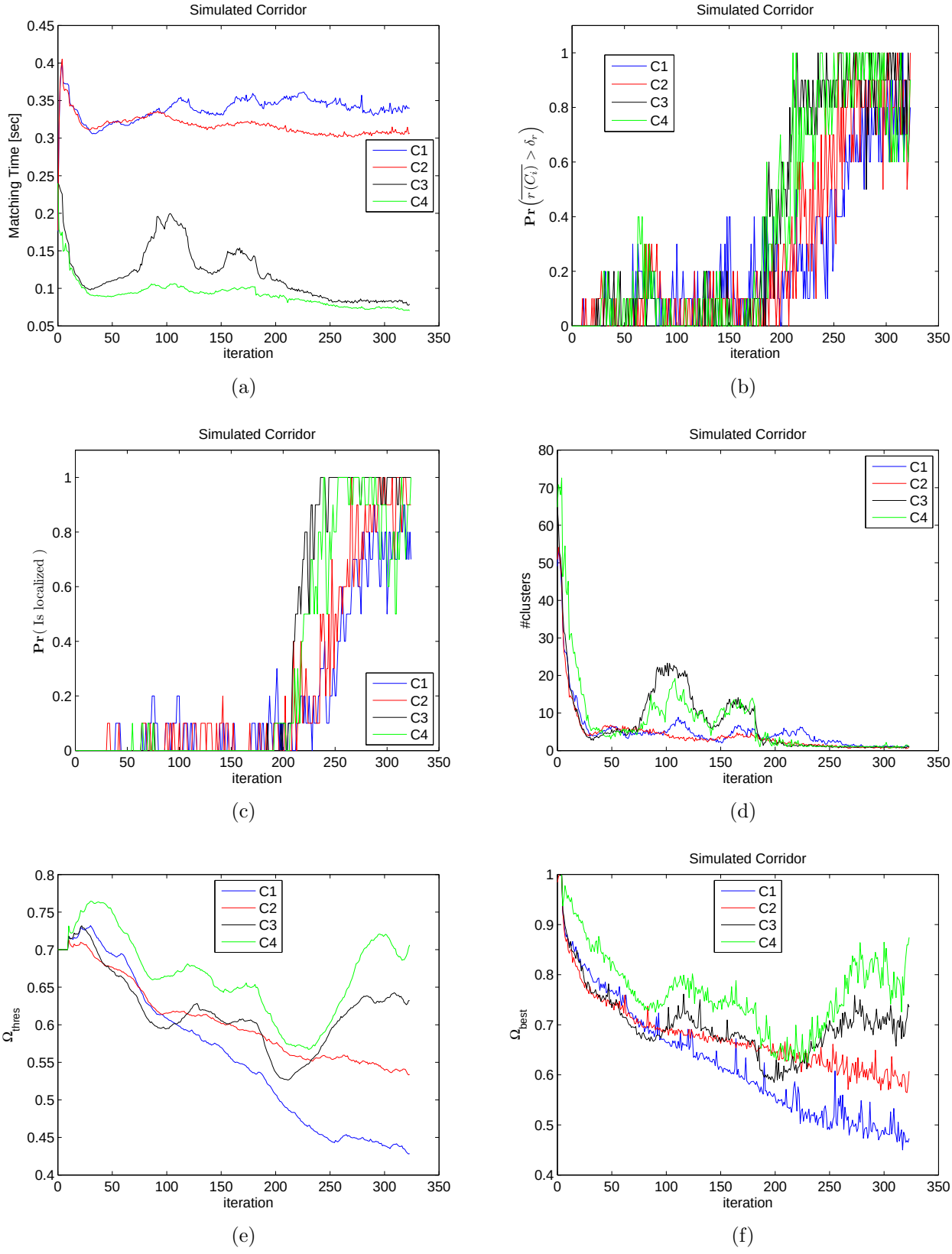
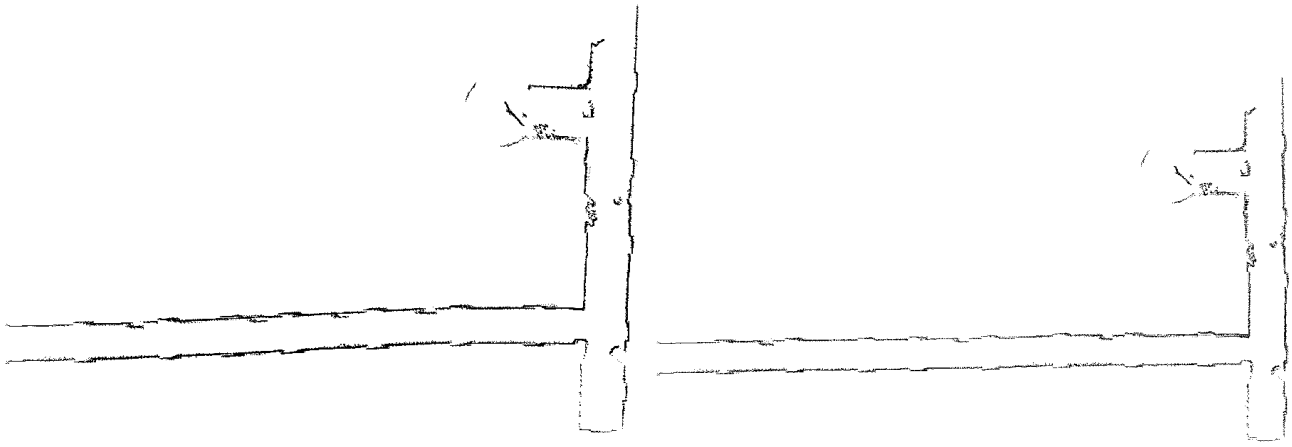
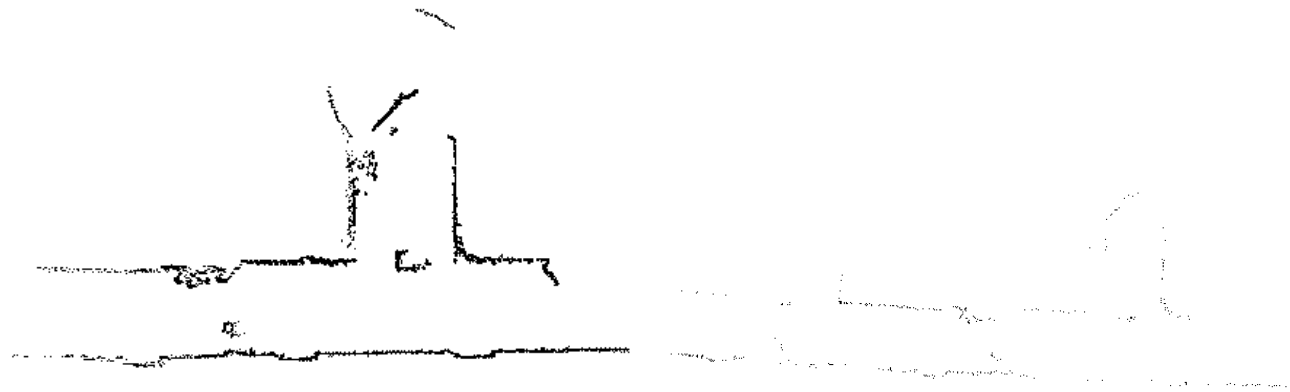


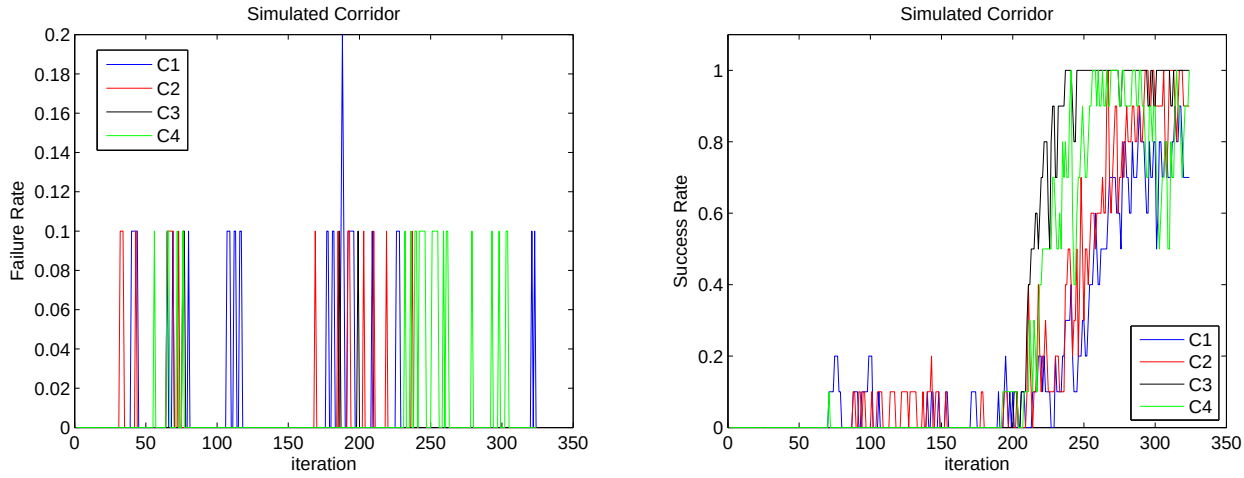
Figure 5.15.: Simulation results - a) Matching time b) Probability that a good cluster has been generated c) Probability that the localization constraints are satisfied d) Number of clusters e)The matching quality threshold f) The best matching quality



(a) (b)



(c) (d)



(e) (f)

Figure 5.16.: Simulation results - a) The complete local map b) The complete local map after application of the gradient descent approach for map correction c) The local map with limited size d) Data reduced local map

5.7.2. Comparison of MML and MCL

The proposed localization method was compared to Monte Carlo Localization (MCL) in three standard scenarios of the last section namely the basement floor of the iRP robotics laboratory, its second floor, and building 079 of the University of Freiburg. Furthermore, MCL was applied to global localization in the MIT workspace but the robot executes another path as in the previous MIT experiment. MCL was configured as follows. The robot was equipped with a SICK LMS laser range-finder in each experiment. Thus, a laser model was used similar to the one proposed by Thrun *et al.* [32]. In order to save computation time, only 18 beams per observation were employed [32]. The expected values necessary to calculate the weights of the particles were computed based on ray-tracing operations sped up by distance images generated from the global map. The cell values constitute the distance to the next obstacle and thus every pixel within the current distance radius can be skipped safely. No look up table was used to precompute the expected values since the sizes of the look up tables distent to a few hundred MB for larger workspaces. The localization constraint of MCL is based on calculating the distance between *expected* particle pose and *best rated* particle pose. More precisely, the best particle $\boldsymbol{x}_{t-1}^{best}$ at time $t-1$ and the current motion command u_t yield an *expected* particle pose $\hat{\boldsymbol{x}}_t = g(\boldsymbol{x}_{t-1}^{best}, u_t)$ at time t where $g(\cdot,\cdot)$ is a non-linear function executing the motion command. MCL provides a best particle $\boldsymbol{x}_t^{best}$ at time t. The distance $d_t = \mid \boldsymbol{x}_t^{best} - \hat{\boldsymbol{x}}_t \mid$ is observed over the last τ_{MCL} time steps employing a sliding window. If the variance of the last τ_{MCL} distances falls below a certain threshold, MCL returns a positive feedback. This is comparable to the third localization constraint of MML. In this experimental setup τ_{MCL} was set to 10. Each of the following experiments compares the ground truth distance of MCL and MML. To this end, the evolution of the best particle and the mean of all particles are considered. Moreover, in the case of MCL, success rate means that the localization constraint must be satisfied *and* the best particle as well as the mean must be closer to the ground truth than a predefined threshold. The experiments will show that the mean converges very smoothly while the pose of the best particle exhibits high frequent jumps. However, exploiting the uncertainty of the best particle may speed localization up since the mean and thus the variance of the particles often require more iterations until convergence. The relation table of MML representing the global map was precomputed for providing compelling computation times. For a fair comparison the computation time of MML includes both the time to update the local map and the matching time.

iRP Robotics Laboratory - Basement

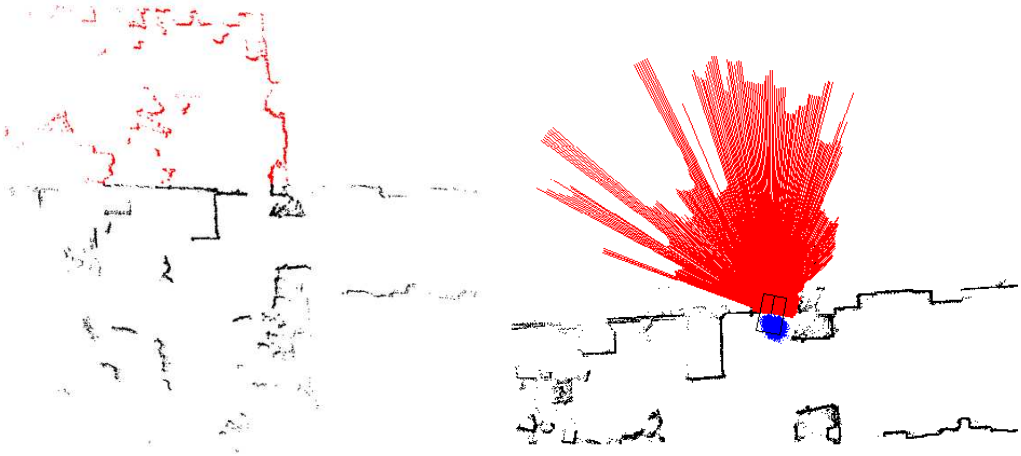
In this scenario 15,000 particles are sufficient to localize the robot. The comparison of MML and MCL is depicted in Fig. 5.18. The computation time of both algorithms is shown in Fig. 5.18a. Obviously, MCL outperforms MML with respect to computation time. The probability of getting a positive localization result can be seen in Fig. 5.18b. MML localizes the robot much faster than MCL. Furthermore, the ground truth error

of MML is very low from the beginning as illustrated by Figs. 5.18c and 5.18d. The success and failure rate are depicted in Fig. 5.18e and 5.18f. Failure rate means that a positive localization result is given and the ground truth errors exceed 50 cm or 20°. In this experiment, the mean and the best particle converge almost simultaneously to the correct robot pose.

iRP Robotics Laboratory - Unknown Area

In this scenario, the behavior of MML and MCL is investigated if the robot enters areas which are not part of the global map. Consequently, localization should become an impossible task while the robot is moving in this regions. Thus, a mechanism which allows for the detection of unknown regions may speed up the localization process since the robot could react quickly and explores other regions of its environment. In Sec. 5.6, a technique was proposed to counteract this problem. The idea is based on the assumption that the last τ' observations do not have correspondences in the global map. Hence, the current matching quality Ω' should very poor. The path of the robot is the orange curve highlighted in Fig. 5.6a. The initial pose is marked by the red circle while the final pose is marked by the blue circle. The red area at the top of the Figure encompasses the unknown region. The robot enters this place at iteration 30 and leaves it around iteration 60. The complete local map after 100 iterations is depicted in Fig. 5.17a. About fifty percent of the map (marked as red points) belong to the unknown place. No computation time is shown since a comparison is out of the scope of this experiment. Fig. 5.19b depicts the matching quality of the last τ' observations. Clearly, Ω' collapses to a poor quality while the robot is moving in the unknown region. Therefore, Ω' provides a good indication whether the robot is moving in information rich areas or not. Fig. 5.19a shows the localization rate. Once the particles are clustered, MCL assumes that the robot pose could be uniquely determined which is not the case and thus MCL provides erroneous localization results while the robot moves in the red area. The particles do not leave the global map but remain at the border where the robot left the mapped workspace. This can be explained by the weight of the particles which is higher in known parts of the laboratory. This phenomenon is illustrated in Fig. 5.17b. On the contrary, the success rate of MCL collapses to zero in the critical part of this experiment while MML is able to provide accurate pose estimation with a considerable probability (cf. Fig. 5.19d). The translational distance to the ground truth is depicted in Fig. 5.19e. The error of MCL exhibits linear characteristics while MML generates some outliers. The rotational distance to the ground truth is shown in Fig. 5.19f. The error generated by MCL increases while the robot stays in the unknown area. The characteristics of MML are strongly influenced by outliers. Therefore, the failure rate of both methods is interesting to see (cf. Fig. 5.19c). The error bounds are 50 cm and 20°. MML clearly detects the outliers since the failure rate is always zero. Opposite to that, the failure rate of MCL is 100 percent while the robot is located in the unmapped area. Note that the average error of MCL is higher after leaving the unknown place due to one trial where the particles did not converge to a cluster when the robot entered the unmapped area. As a result, the filter diverged in this case while

MML always 'recovers' from the critical region, i.e. the high frequent jumps disappear completely.



(a) (b)

Figure 5.17.: a) The complete local map after 100 iterations. The unknown area is marked by the red map points b) The particles do not leave the mapped area because the particles outside gain very low weights.

iRP Robotics Laboratory - First Floor

The first floor of the iRP robotics laboratory is a hard scenario for MCL because the sensor faces a wall made of glass. Thus, depending on the angle of incidence, the laser beams are not reflected. However, the glass appears as a normal wall in the map yielding large discrepancies between expected distance and real measurements. The area is marked by the orange ellipse in Fig. 5.6b. 20,000 particles are employed in this scenario. The computation time of both approaches is compared in Fig. 5.20a. MCL is temporary faster than MML since the ray tracing operations are very fast close to the top right corner of the map (cf. Fig. 5.6b). In contrast, ray tracing consumes more computational resources if the robot faces the long corridor in the second half of its path. The rate of positive localization steps is depicted in Fig. 5.20b. Obviously, MML localizes the robot much faster than MCL since MML does not suffer from the invisible wall problem mentioned above. Again, this does not mean that the localization is accurate. The algorithm has just a high level of confidence that the pose could be determined uniquely enough. However, the ground truth error shown in Fig. 5.20c and 5.20d indicate that the estimated pose indeed is very precise from the beginning. The best particle and the mean require much more iterations until convergence due to the glass wall faced by the robot at the beginning of the localization. Additionally, Fig. 5.20c illustrates that the pose of the best particle converges faster than the mean but due to the queue controlled by τ_{MCL} the number of localization steps is only marginally decreased (cf. Fig. 5.20e). The success and failure rate are depicted in Fig. 5.20e and 5.20f. The error bounds are 50 cm and 20° again. Both figures indicate that the distance of the

mean to the ground truth is more stable than the distance of the best particle. The success rate of MML is excellent from the beginning. As as summary, MML outperforms MCL with respect to the number of required motions until the robot is localized. The performance with respect to computation time is fairly balanced.

fr079

This workspace demands 30,000 particles for a reliable and accurate localization. Fig. 5.21a depicts the computation time of MML and MCL. The filter update time decreases during convergence since the robot is still located in a small room. At iteration 60, the robot enters the corridor in the middle of the map which remarkably increases the update time. The computation time of MML is more stable but higher compared to the previous scenario because the scan matcher consumes more resources. The reason is that the quality of the odometry is worse which must be dealt with by the scan matcher. The localization rate of MML outperforms MCL. The error bounds are defined as 50 cm and 20° in this experiment. The success rate and the failure rate are highlighted in Figs. 5.21e and 5.21f. The failure rate implies that the error generated by MML is always smaller than 50 cm and 20°, respectively. The ground truth error is depicted in Fig. 5.21c and 5.21d. Again, the MML result is very accurate from the beginning while MCL need almost 20 iterations until the best particle is on the same level of precision. The success rate of the MCL mean again outperforms MML after convergence of the filter. However, the success rate of MML is on an excellent level at a very early stage of the experiment.

MIT Killian Court

The MIT dataset is the most challenging scenario for both MML and MCL. The path executed by the robot is shown in Fig. 5.6e. An amount of 120,000 particles is required in this area due to its impressive dimensions. The computation time is depicted in Fig. 5.22a. MML clearly outperforms MCL with respect to computation time since MML is about five times faster than MCL by reason of the large number of particles. The probability that the system satisfies the localization constraints is shown in Fig. 5.22b. MCL requires more than 40 filter updates until an acceptable level is achieved. On the contrary, MML localizes the robot much earlier. Note that the localization probability of MCL oscillates conspicuously even after convergence of the filter. The best particle *jumps* within the cluster which may be explained by sensor readings gathered in a corridor environment. The success rate of MML (the ground truth error must be smaller than 100 cm/20°) increases very early as shown by Fig. 5.22e. Moreover, no erroneous localization occurred as Fig. 5.22f proves. Indeed, if the localization constraints are satisfied, the average errors of MML are 23.27 cm and 1.27 °. Since the failure rate of MML is always zero, the success rate equals the localization probability. In contrast, the performance of MCL is rather poor for both the best particle and the mean which is not surprising since the best particle is used in the localization constraint. However,

the success rate of the mean converges smoothly to a 100 percent level close to iteration 90. The average ground truth error is depicted in Fig. 5.22c and 5.22d. Obviously, the average error of MCL including all localization trials is smoother than the average error of MML. This is clear because after convergence of the filter, the particles are clustered close to the ground truth pose while pRANSAM might generate matching results far from the true robot pose. This implies that MCL is more suitable for tracking given that global localization was successful. This statement is supported by the results of all other scenarios. Fig. 5.22c, Fig. 5.22d, and Fig. 5.22e perfectly illustrate that the best particle may be a preferable choice for exploitation in a localization constraint since its pose converges much faster than the mean. Simultaneously, the disadvantage is obvious since the best particle exhibits a considerable distance to the ground truth when the robot moves in a corridor environment with few corners.
As a conclusion for this experiments, both techniques have advantages and drawbacks and thus their performances are fairly balanced but the localization is faster employing MML. Consequently, MML is an excellent mean to initialize a tracking algorithm like MCL. However, no MCL configuration was found which facilitates a successful localization for the robot path depicted in Fig. 5.6f. Due to the large distance which needs to be covered by the robot until localization is possible, filter divergence was always observed. This phenomenon may be explained by many dynamic obstacles crossing the field of view of the sensor while the robot passes the infinite corridor (cf. Fig. 5.14e). As a result, the *Markov Assumption* (cf. App. A.3) is violated and the filter diverges. Consequently, more sophisticated techniques for non-static environments as presented in the literature are required for MCL. Note that MML indeed yields excellent results although its configuration must be chosen more carefully than in smaller workspaces. This underlines the effectiveness of the proposed method compared to standard localization techniques.

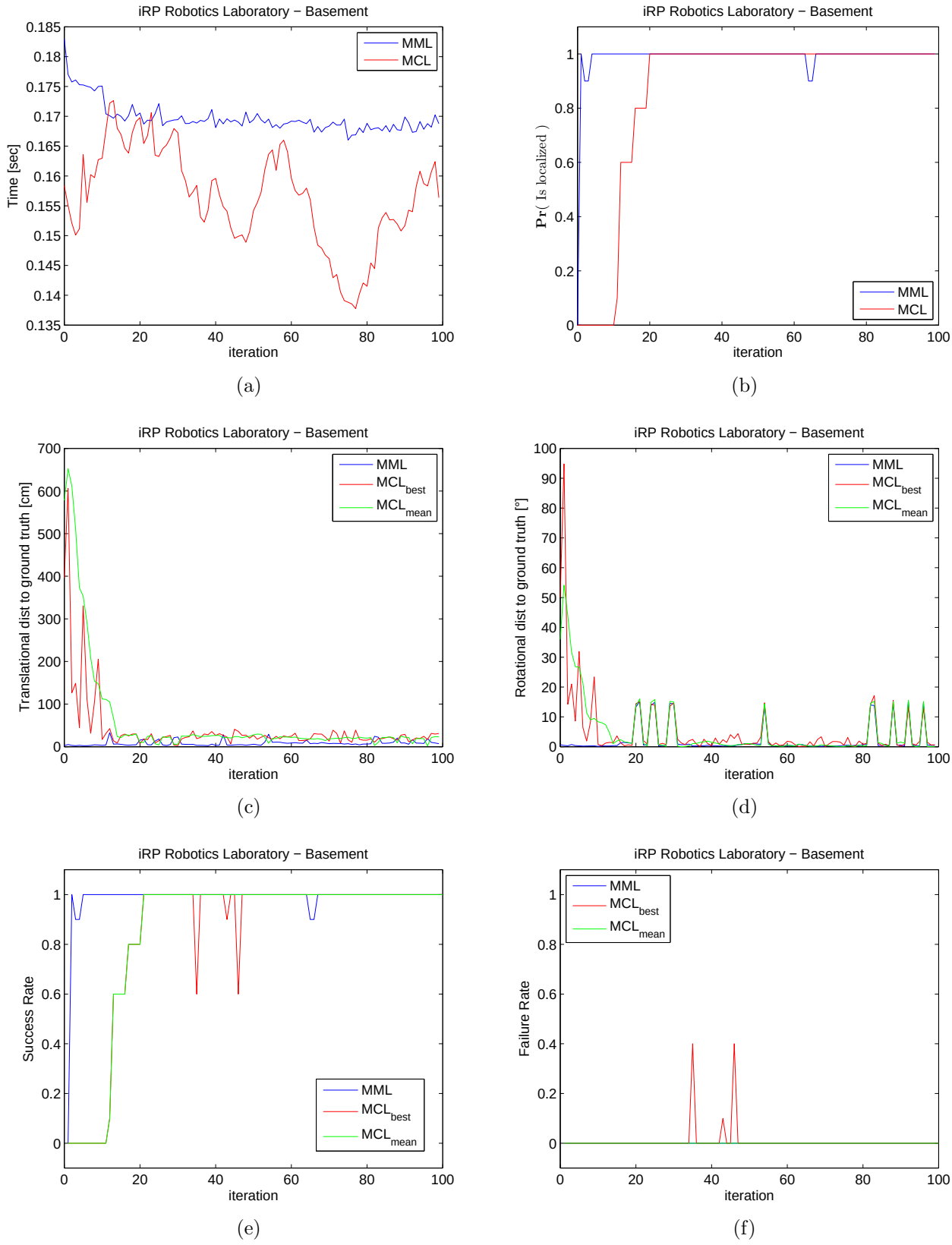


Figure 5.18.: Comparison of MML and MCL in the basement scenario - a) (Filter) update time b) Probability that the localization constraints are satisfied c) Translational distance to ground truth d) Rotational distance to ground truth e) Success rate f) Failure rate

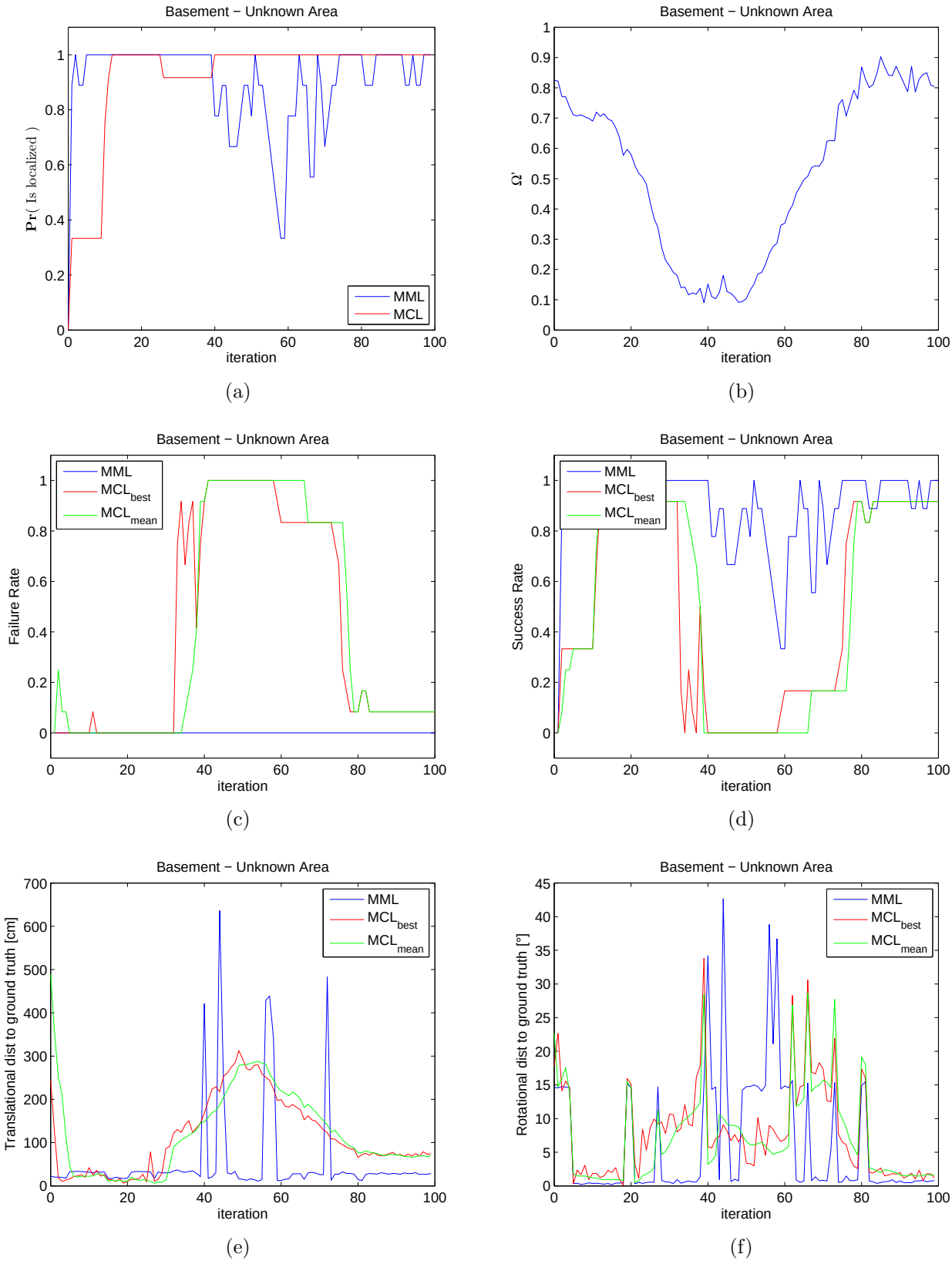


Figure 5.19.: Comparison of MML and MCL in some unknown parts of the basement - a) Probability that the localization constraints are satisfied b) Portion of the last τ' observations which are in contact with the global map c) Failure rate d) Success rate e) Translational distance to ground truth f) Rotational distance to ground truth

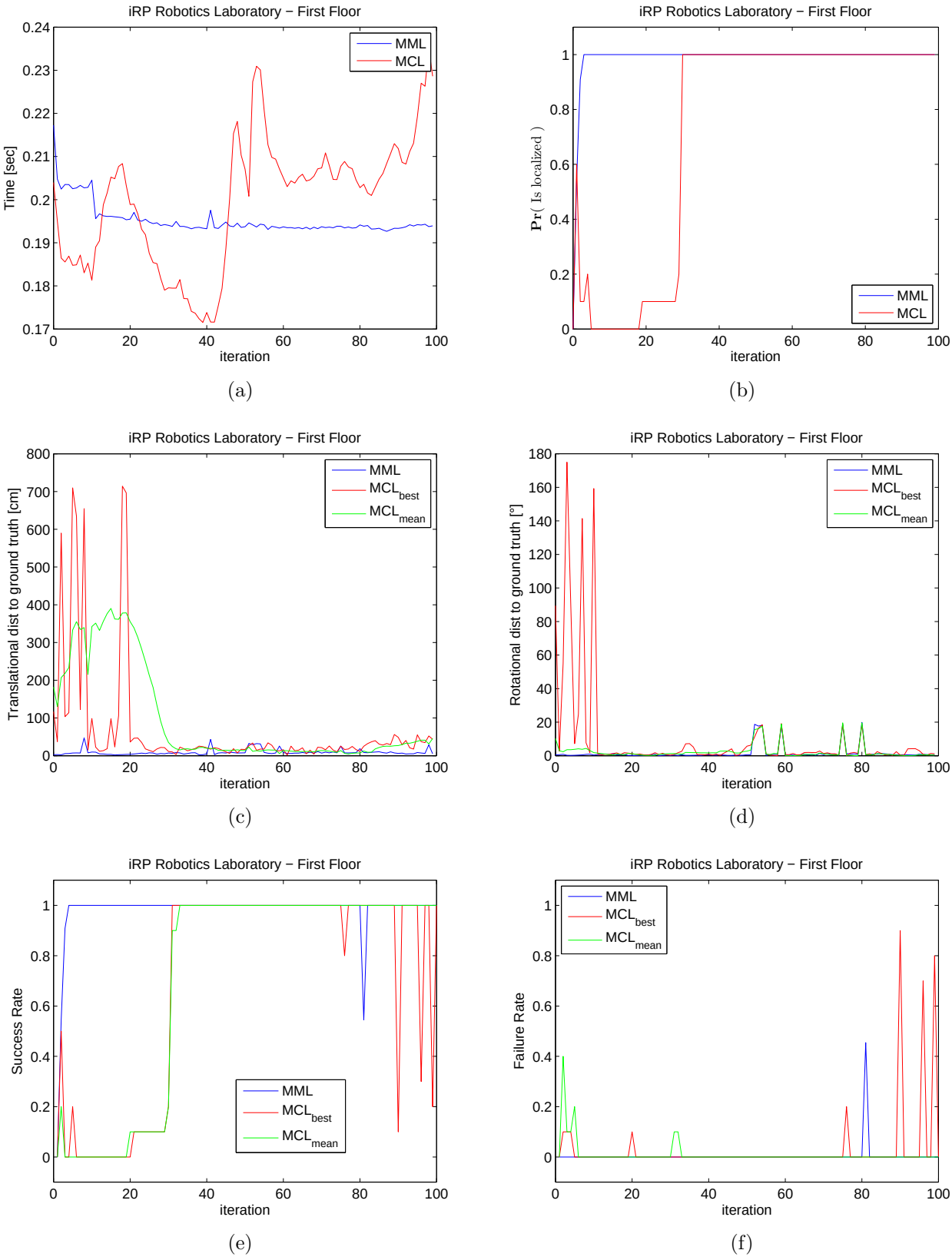


Figure 5.20.: Comparison of MML and MCL in first floor of the iRP robotics laboratory - a) (Filter) update time time b) Probability that the localization constraints are satisfied c) Translational distance to ground truth d) Rotational distance to ground truth e) Success rate f) Failure rate

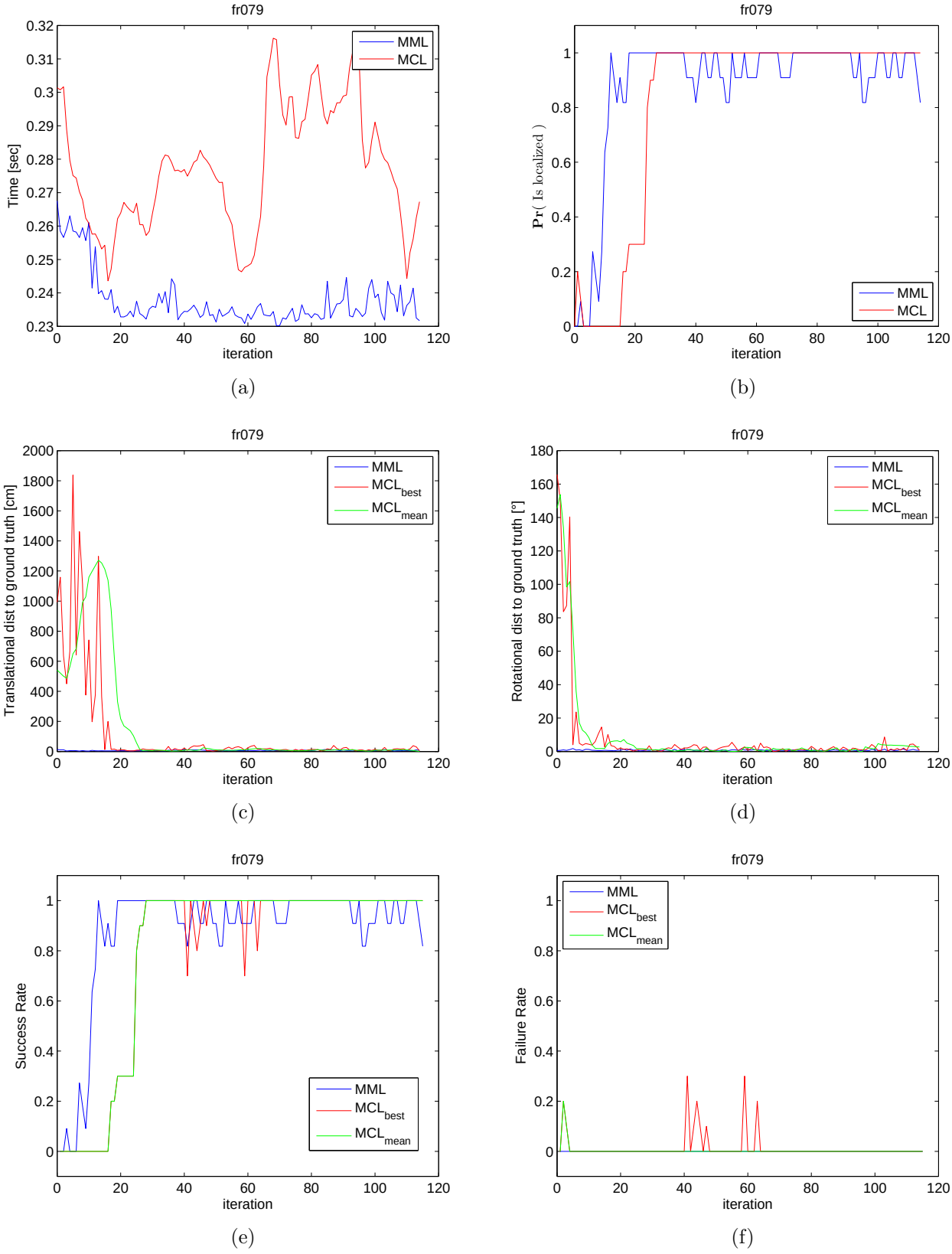


Figure 5.21.: Comparison of MML and MCL in building 079 of the University of Freiburg - a) (Filter) update time time b) Probability that the localization constraints are satisfied c) Translational distance to ground truth d) Rotational distance to ground truth e) Success rate f) Failure rate

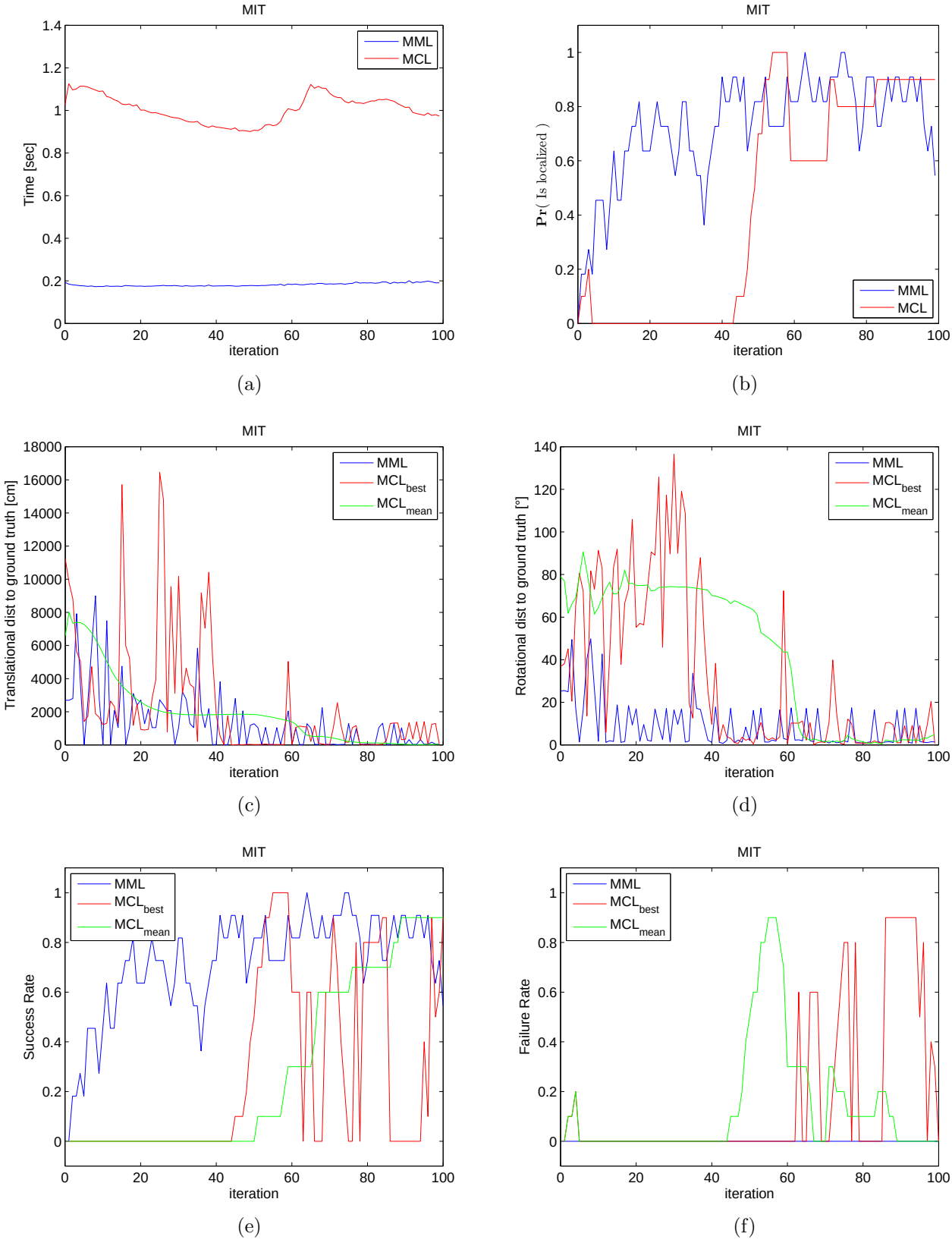


Figure 5.22.: Comparison of MML and MCL in the Boston MIT Killian Court - a) (Filter) update time time b) Probability that the localization constraints are satisfied c) Translational distance to ground truth d) Rotational distance to ground truth e) Success rate f) Failure rate

5.8. Conclusion

An efficient and reliable algorithm (MML) for global localization was presented in this chapter. The underlying idea is to compute a local map incrementally and match the local map to a global model employing a highly parallelized technique for global point cloud matching named pRANSAM. The scan matcher as well as pRANSAM were intensively discussed in previous chapters. A method was explained for postprocessing the hypotheses resulting from the matching operation including clustering and cluster weighting. Moreover, mathematical constraints were proposed identifying an accurately isolated robot pose. Another aspect is the detection of unmapped areas since localization becomes impossible if the robot moves in such regions. To this end, a heuristic for the identification of unknown places was discussed based on the assumption that local map points representing unmapped areas do not have corresponding points in the global map. An obvious drawback of the localization principle is the well-known fact that local maps may exhibit inconsistencies over time since a pure scan matcher is employed for local map computation. In order to counteract this problem, two alternative techniques were proposed. The first option is to keep the size of the local map on a fixed level diminishing the effect of map inconsistencies. The second option is to apply a correction step which involves a gradient descent routine. The idea is based on the heuristic that the regions around the computed hypotheses are of similar geometrical shapes. Further important aspects refer to runtime issues and methods were discussed to decrease the computational effort during localization. Finally, experimental results demonstrated the characteristics of the algorithm comparing several configurations settings. Additionally, the performance of the proposed scheme was compared to traditional Monte Carlo Localization. Experiments revealed that MML outperforms standard MCL during the localization stage. However, due to the RANSAC-based nature of MML, it is not very suitable for tracking once the correct robot pose is determined. Thus, MML is a proper mean to initialize a tracking algorithm, e.g. MCL with very few particles, Markov Localization, or an EKF [32]. In summary, MML showed very promising results which renders it worthwhile for further investigations and improvements.

References

[1] S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *Pattern Recognition (DAGM 2006)*, pages 718–728, 2006.

[2] R. Westphal. *Sensor-Based Surgical Robotics: Contributions to Robot Assisted Fracture Reduction.* Fortschritte in der Robotik. Shaker, 2007.

[3] R. Iser and F. Wahl. Building local metrical and global topological maps using efficient scan matching approaches. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1023–1030, 2008.

[4] S. Winkelbach and S. Molkenstruck. David 3d scanner. ⟨http://www.david-laserscanner.com⟩ [date: 07/06/2009]. Internet, 2009.

[5] R. Iser, J. Spehr, S. Winkelbach, and F. Wahl. Mobile robot localization using the fast random sample matching approach. In *Proc. of Robotik 2008*, pages 163–166, 2008.

[6] S. Se, G. D. Lowe, and J. J. Little. Global localization using distinctive visual features. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 226–231, 2002.

[7] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of IEEE International Conference on Computer Vision*, pages 1150 –1157, 1999.

[8] P. Ju-Hong, K. Soohwan, D. Nakju, and P. Sung-Kee. Vision-based global localization based on a hybrid map representation. In *International Conference on Control, Automation and Systems (ICCAS)*, pages 1104 –1108, October 2008.

[9] K. Tanaka and E. Kondo. Incremental ransac for online vehicle relocation in large dynamic environments. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1025–1030, 2006.

[10] K. Tanaka and E. Kondo. Towards constant-time robot localization in large dynamic environments. In *Proc. of IEEE Int. Conf. on Networking, Sensing and Control*, pages 113–118, 2006.

[11] D. Nister. Preemptive ransac for live structure and motion estimation. In *Proc. of IEEE International Conference on Computer Vision*, volume 1, pages 199 –206, October 2003.

[12] T. Ueda and K. Tanaka. On the scalability of robot localization using high-dimensional features. In *Proc. of the Int. Conf. on Pattern Recognition*, pages 1–4, 2008.

[13] G. Mori, S. Belongie, and J. Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832 – 1837, November 2005.

[14] S. Arya, D. M. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45:891–923, November 1998.

[15] K. Saeki, K. Tanaka, and T. Ueda. LSH-RASNAC: An incremental scheme for scalable localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3523–3530, 2009.

[16] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of the 25th Very Large Database Conference (VLDB)*, pages 718–728, 1999.

[17] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[18] S. Lamrous and M. Taileb. Divisive hierarchical k-means. In *Proc. of the IEEE International Conference on Computational Intelligence for Modelling Control and Automation*, pages 18–18, 2006.

[19] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier. Morgan Kaufman, 2005.

[20] A. Burguera, Y. Gonzalez, and G. Oliver. Probabilistic sonar scan matching for robust localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3154–3160, April 2007.

[21] R. Iser, A. Martens, and F. Wahl. Localization of mobile robots using incremental local maps. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 4873–4880, 2010.

[22] J. Bollinger. *Bollinger on Bollinger Bands*. McGraw-Hill, 2001.

[23] D. Meyer-Delius and W. Burgard. Maximum-likelihood sample-based maps for mobile robots. In *Proc. of the European Conference on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

[24] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence Journal*, 128(1-2), 2001.

[25] D. Fox. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, 2003.

[26] P. Pfaff, C. Plagemann, and W. Burgard. Gaussian mixture models for probabilistic localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 467 –472, May 2008.

[27] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Monte carlo localization in outdoor terrains using multilevel surface maps. *J. Field Robot.*, 25:346–359, June 2008.

[28] J. Blanco, J. González, and J. A. Fernández-Madrigal. Optimal filtering for non-parametric observation models: Applications to localization and slam. *The International Journal of Robotics Research*, 2010.

[29] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. *Proceedings of the IEEE*, 92(3):469 – 484, March 2004.

[30] M. Nieuwenhuisen, J. Stückler, and S. Behnke. Improving indoor navigation of autonomous robots by an explicit representation of doors. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 4895 – 4901, May 2010.

[31] P. Pfaff, C. Stachniss, C. Plagemann, and W. Burgard. Efficiently learning high-dimensional observation models for monte-carlo localization using gaussian mixtures. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3539 –3544, September 2008.

[32] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. The MIT Press, 2005.

Chapter 6

Summary and Discussion

In this thesis a novel RANSAC-based localization framework was proposed. The underlying idea was to compute a local map which was matched to a global one for pose estimation. The matching routine provided a set of hypotheses postprocessed in subsequent steps in order to resolve structural ambiguities. In this chapter the algorithms introduced in this thesis and its results are summarized. Furthermore, open questions and issues subject to further improvements are discussed.

The construction of the local maps required a reliable scan matcher introduced in Chap. 2. The proposed method was based on a particle filter suitable for tracking the robot pose, e.g. during path execution. For matching the current observation to previous measurements, odometry data were used to get a rough estimate of the new robot pose. Then a set of particles was spread around the expected pose by means of a Gaussian distribution. The improvement of the algorithm which facilitates accurate local map building affected the resampling step. Instead of resampling only once it was repeated several times until an abortion constraint was satisfied. This criterion was based on the effective sample size which served as an indicator how reliable the particles are distributed for a correct matching. The covariance of the Gaussian distribution was decreased in each iteration in order to cluster the samples as densely as possible. Compared to two state-of-the-art scan matching techniques, i.e. ICP and PSM, the proposed approach required at least one order of magnitude more computation time. In contrast, consistent maps could be computed without any optimization strategy, e.g. from the popular Intel Research Laboratory data set while ICP and PSM yielded poor results. Thus, the proposed method is an excellent means to compute consistent maps of small to medium size environments. Scan matchers discussed in the literature which exhibit comparable global accuracy are rare. The Normal Distributions Transform (NDT) of Biber *et al.* [1] showed the most competitive results. It was investigated by other researchers but no explicit loop closing experiments were conducted [2, 3, 4]. Therefore, despite of the computational burden, the proposed technique remains a challenging tool if global map accuracy is an important matter as it is the case in this thesis. However, the algorithm was optimized for the usage of accurate laser range-finders. Grid-cells were updated employing an end point model, i.e. sensor noise was not modeled in the current implementation. At the end of Chap. 2.3 a technique from the literature was

discussed which may be beneficially combined with the scan matcher for the integration of noisy sensors like ultrasonic. The capability of building accurate local maps with low-cost sonar sensors would be an essential step towards using those sensors in the overall localization approach.

A SLAM method to generate the global map was described in Chap. 3 which offered more sophisticated means to assure map consistency. The approach was divided into a front-end and back-end which is a common structure of graph-based SLAM techniques. The front-end established a topological structure of the global map which was represented as a graph with nodes and edges. For this purpose, a set of local map fragments was computed applying the scan matcher presented in Chap. 2. The local maps were connected by samples drawn from Gaussian distributions where each sample represents a hypothesis of the transformation between consecutive map fragments. Thus, the nodes of the graph were metrical maps while the edges carried information about the relative transformation of their nodes. Active loop closing was performed using the same matching strategy employed for global localization. If a loop closing event was detected, the matching routine yielded a hypothesis for the correct alignment of the current map fragment and the corresponding one. This hypothesis was a fundamental mean to asses global map solutions during the optimization routine. Nodes were chosen as loop closing candidates based on an estimation of the robot pose with respect to the base frames of the local maps. After loop closing, Dijkstra's shortest path algorithm was applied to compute a sequence of nodes representing the loop under the side condition of a shortest route. Ant Colony Optimization (ACO) constituted the back-end of the SLAM algorithm. Ants were applied to explore the sub-graph provided by Dijkstra where each single edge sequence represented an individual solution to the SLAM problem. The motivation of using an ACO algorithm as a back-end was twofold. Predominantly, it was interesting to see how a swarm intelligence technique successfully applied to the Traveling Salesman Problem (TSP) could be integrated into a SLAM approach. The second motivation was due to the fact that gradient-descent methods and non-linear least-squares techniques may get stuck in local minima yielding suboptimal map results. Experiments demonstrated that ACO was able to construct accurate maps. However, a comparison to a Levenberg-Marquardt (LM) optimization revealed that the local minimum problem is negligible for the map computation, i.e. the human eye perceived no remarkable difference. Since LM optimization normally converges faster and yields a higher accuracy, it is preferable to ACO. Nevertheless, investigations showed that situations exist were ACO yields a better map quality than LM optimization. The main aspect which requires further enhancement pertains the data association step for loop closing. The above mentioned map matching strategy often exhibits poor results due to ambiguous geometrical structures. Moreover, the concept of the borders which separate the local maps needs to be improved for SLAM in open places like halls or in outdoor scenarios where local maps frontiers are hard to define.

A parallelization of the RANSAM method (pRANSAM) was introduced in Chap. 4 as an efficient means to match local and global maps. Registration of the point triples to the relation tables was distributed among several threads. Each thread performed

point sampling, collision detection, and hypotheses evaluation. A super linear speedup was observed for the dual core version of pRANSAM which may be due to improved cache efficiency. A linear speedup was achieved when applying pRANSAM on a quad core machine. The original RANSAM uses a hard contact criterion for point to point correspondences, i.e. a point has contact to its corresponding point or not. The computational burden could be decreased by introducing a fuzzy contact criterion yielding an additional speedup while maintaining subjective matching quality. Matching 2D laser scans to maps of typical indoor environments was executed in less than 20 ms which qualified pRANSAM as an appropriate mean for local map matching in the localization framework. Furthermore, an optimal allocation of the threads to the relation tables could be found. A theorem was derived which claims that given T threads the probability of finding correct point to point correspondences reaches its maximum if $\frac{T}{2}$ threads draw from the first point set and $\frac{T}{2}$ threads draw from the second point set. Hence, the thread allocation did not depend on the cardinalities of the point clouds. Consequently, pRANSAM is very easy to configure for practical applications. In particular, the thread distribution did not need to be adapted dynamically to growing local map size during global localization. Moreoever, pRANSAM can be easily employed in all applications presented in the thesis of Winkelbach [5] yielding a substantial speed-up depending solely on the number of CPU-cores.

The scan matcher and pRANSAM were fused into the localization framework described in Chap. 5. It was named as Map Matching Localization (MML). Originally, RANSAM was configured to provide a single best hypothesis if either a quality threshold was exceeded or a maximum number of iterations was reached. This was insufficient for localization since one hypothesis did not provide any information about structural ambiguity, i.e. whether the robot pose could be uniquely determined. Thus, given a quality threshold pRANSAM run for a maximum number of iterations collecting each hypothesis which exceeded the quality threshold. As a result, pRANSAM was likely to deliver many hypotheses even if the robot pose was 'unique'. Therefore, the hypotheses were recursively clustered using a hierarchical K-means algorithm. The resulting clusters were assessed by means of a rating function, e.g. considering the average matching quality of the portion of hypotheses which belong to the cluster. Subsequently, the weights were normalized in order to define a threshold which determines promising pose candidates. Hence, the cluster rating was chosen as a first localization constraint. Indeed, in many office environments this constraint was sufficient to exclude spurious hypotheses but additional constraints were required to detect erroneous matching results. Therefore, the hypotheses were 'tracked' using odometry data and the distance of the expected pose to the current hypothesis was computed constituting a second constraint. Finally, the 'uncertainty' about the distances of the time steps was observed which defined a third constraint. Putting this together yielded a robust criterion to determine whether localization was accurate and reliable. Concerning runtime issues, it was proven that the matching time of the local map could be further decreased by precomputing the relation table of the global map, i.e. T threads simultaneously fill the relation table of the local map. Furthermore, a strategy was proposed to reduce the amount of data matched to the global map. Local map divergence is an obvious drawback of the pro-

posed localization algorithm. Hence, two alternatives to counteract this problem were discussed. The first one was to apply a gradient descent correcting the local map. The idea was based on the assumption that the local regions around the hypotheses generated by pRANSAM are of similar geometrical shapes. The second option was to simply keep the local map size constant, i.e. discarding old map information. Except from local map consistency a further advantage was a nearly constant matching time which is an important matter for real time applications. Finally, intensive experiments demonstrated the robustness and accuracy of the proposed localization method. The intention of the experiments was twofold. First, several configuration sets of MML were compared in order to show the effect of the described improvements, e.g., precomputation of the relation tables, application of the Bollinger Bands etc.. It was shown that the localization is robust and fast even in large scale environments like the MIT Killian Court. The second intention was to compare MML with standard Monte Carlo Localization (MCL) as a popular representative of Bayes filter approaches. The performance of MLC and MML was fairly balanced. MML required less iteration in order to isolate the correct robot pose. This constitutes the superior advantage of MML towards filter-based localization since MML does not suffer from divergence. Therefore, the parameters determining the localization speed could be chosen more optimistically while maintaining similar or better accuracy and robustness. Additionally, MML was faster in large scale environments since a considerable amount of particles is required to avoid the particle depletion problem. On the contrary, MML is not very suitable for tracking due to the RANSAC-based structure of the algorithm. Consequently, a combination of both methods would be an interesting option due to their complementary strengths, i.e. MML performs global localization and initializes MCL for further tracking. However, a comparison with advanced implementations of MCL as described in the Chap. 1.1 would be very interesting since more sophisticated sensor models may reduce the required number of particles and increase the robustness of the filter. Compared to state-of-the-art RANSAC localization [6, 7, 8, 9], MML has several superior advantages. First, no features need to be extracted from the sensor measurements. Consequently, MML can be used in almost arbitrary environments. Second, the localization constraints are of relevant practical benefit since they entail a robust measure when localization is completed. This matter was completely ignored in the literature. Third, recent publications did not discuss the problem of local map inconsistency. Instead, a 'SLAM method on the local level' was silently assumed. Finally, this is the first study validating the advantages and drawbacks of RANSAC-based localization compared to MCL. Future work will focus on three aspects.

- The main drawback of the proposed approach is the need for local maps which may exhibit inconsistencies since their sizes increase constantly with each iteration. Two methods were discussed to counteract this problem. The first one applied a gradient-descent technique to correct the local map given the best rated hypothesis generated by pRANSAM. However, experiments showed that this approach might yield poor results, e.g. if moving people cross the sensor. In this case, the gradient descent is likely to fail due to bad point to point correspondences. Hence, this technique requires further improvements for a robust application. A second option

was to bound the size of the local map. A positive side-effect was a nearly constant matching time per iteration but this idea does not scale very well with the size of the workspace. Depending on the environment, even a local map of constant size may be too large to guarantee consistency. Consequently, this matter is subject of further research. A possible solution could be the application of an advanced SLAM technique to build the local maps. But it is important that the algorithm is capable of solving the *full* SLAM problem since consistent local maps need to be guaranteed without any loop closing.

- Localization was performed in 2D but several SLAM techniques exist nowadays which provide 3D maps. Therefore, it would be interesting to investigate the performance of the presented method using 3D maps. If another map representation than point clouds is employed, pRANSAM needs to be adapted. Moreover, the feasibility of the proposed scan matcher in 3D applications is questionable due to its computational costs and thus an adaption or even a replacement of the scan matcher has to be considered.
- Throughout this thesis, laser range-finders were used to compute the maps. Unfortunately, cheaper sensors like cameras or sonar are more appealing, especially in industrial applications. Hence, improvements which allow for the employment of low cost sensors are very interesting. Considering sonar, special techniques need to be devised which facilitate the computation of accurate local maps under the presence of remarkable sensor noise. Therefore, cameras as very information rich sensors may be more attractive to integrate in the approach proposed in this thesis.

As a summary, a robust localization framework was presented in this thesis which provides accurate pose estimations. The overall characteristics exhibit competitive performance with respect to existing algorithms capable of global localization. Although many issues are worthwhile for further enhancements and investigations, the novel technique constitutes an excellent basis for real applications.

References

[1] P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2743 – 2748, October 2003.

[2] E. Takeuchi and T. Tsubouchi. A 3-d scan matching using improved 3-d normal distributions transform for mobile robotic mapping. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3068 – 3073, oct. 2006.

[3] L. Jinliang, B. Jihua, and Y. Yan. Study on localization for rescue robots based on ndt scan matching. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pages 1908 –1912, June 2010.

[4] C. Ulas and H. Temeltas. A fast and robust scan matching algorithm based on ml-ndt and feature extraction. In *Mechatronics and Automation (ICMA), 2011 International Conference on*, pages 1751 –1756, August 2011.

[5] S. Winkelbach. *Das 3d-Puzzle-Problem*. Fortschritte in der Robotik. Shaker, 2006.

[6] K. Tanaka and E. Kondo. Incremental ransac for online vehicle relocation in large dynamic environments. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1025–1030, 2006.

[7] K. Tanaka and E. Kondo. Towards constant-time robot localization in large dynamic environments. In *Proc. of IEEE Int. Conf. on Networking, Sensing and Control*, pages 113–118, 2006.

[8] T. Ueda and K. Tanaka. On the scalability of robot localization using high-dimensional features. In *Proc. of the Int. Conf. on Pattern Recognition*, pages 1–4, 2008.

[9] K. Saeki, K. Tanaka, and T. Ueda. LSH-RASNAC: An incremental scheme for scalable localization. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3523–3530, 2009.

Appendix A

Monte Carlo Localization

The localization algorithm proposed in this thesis is compared to the well known Monte Carlo Localization (MCL) with respect to computation time per iteration, accuracy, and success rate (cf. Sec. 5.7.2). The principle of MCL is explained in the following sections.

A.1. Bayes Filter

The Bayes filter is a standard mean to recursively estimate the state of the robot based on a prior estimation, the active control input, and current measurement. The basic update equations are given in Alg. 4. The prior at time $t-1$ is denoted as $bel(\boldsymbol{x}_{t-1})$, i.e. it constitutes a *belief* over the space of all possible states. The control input and the measurement are denotes as u_t and z_t respectively. Line 2 iterates over all states

Algorithm 4 Bayes Filter

1: **procedure** UPDATE($bel(\boldsymbol{x}_{t-1}), u_t, z_t$)
2: **for all** $\boldsymbol{x}_t$ **do**
3: $\overline{bel}(\boldsymbol{x}_t) = \int p(\boldsymbol{x}_t \mid u_t, \boldsymbol{x}_{t-1}) bel(\boldsymbol{x}_{t-1}) d\boldsymbol{x}_{t-1}$
4: $bel(\boldsymbol{x}_t) = \eta p(z_t \mid \boldsymbol{x}_t) \overline{bel}(\boldsymbol{x}_t)$
5: **end for**
6: return $bel(\boldsymbol{x}_t)$
7: **end procedure**

$\boldsymbol{x}_t$ at time t. The so called *prediction* step is performed in line 3 by integrating the product of two probability distributions. The first factor is the prior $bel(\boldsymbol{x}_{t-1})$ and the second probability distribution represents the transition from state $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$ given the control input u_t. Typically, a state $\boldsymbol{x}_t$ denotes a robot pose at time t and u_t is a motion command. Thus, the probability distribution $p(\boldsymbol{x}_t \mid u_t, \boldsymbol{x}_{t-1})$ is the *motion model* of the robot. Line 4 conducts the *correction* step of the update procedure by multiplying the predicted state with the measurement model. Therefore, the correction step is often called *measurement update*. Finally, the posterior $bel(\boldsymbol{x}_t)$ is returned. Note that the basic form of the Bayes filter is not feasible for practical applications since the integral in line

3 must be calculated in a closed form. Another option is that the state space is discrete and finite. Consequently, the integral becomes a sum but finite state spaces are not realistic assumptions. Hence, other implementations of the Bayes Filter for continuous spaces were devised in the past. *Gaussian filters* are a traditional family of recursive state estimators. The Kalman filter, the Extended Kalman filter, the Unscented Kalman filter, and the Information filter are popular examples of algorithms belonging to this class of recursive state estimation techniques. A second family constitute *Nonparametric filters*, i.e. the probability distributions are not represented in closed form expressions. The Histogram filter and the Particle filter are well known members of this family.

A.2. Particle Filter

As stated above, the particle filter is a nonparametric version of the Bayes filter. This implies that the posterior is approximated by a discrete and finite set of hypotheses, i.e. so called *samples*. The sample set

$$\mathcal{S} = \left\{ \left\langle \boldsymbol{s}^{[i]}, w^{[i]} \right\rangle \mid 1 \leq i \leq M \right\}$$

consists of M weighted elements where $\boldsymbol{s}^{[i]}$ represents the state vector of sample i and $w^{[i]}$ denotes its weight. The weights are non-zero values and sum up to 1. Consequently,

$$bel(\boldsymbol{x}) \sim \sum_{i=1}^{M} w^{[i]} \delta(\boldsymbol{s} - \boldsymbol{s}^{[i]})$$

where $\delta(\cdot)$ is the dirac pulse. One can prove that the samples constitute the true posterior if $M \to \infty$. The particle filter is summarized in Alg. 5. The input of the

Algorithm 5 Particle Filter

1: **procedure** UPDATE($\mathcal{S}_{t-1}, u_t, z_t$)
2: $\quad \overline{\mathcal{S}}_t = \mathcal{S}_t = \emptyset$
3: $\quad$ **for** $m = 1$ to M **do**
4: $\quad\quad$ draw $\boldsymbol{s}_t^{[m]} \sim p(\boldsymbol{s}_t \mid u_t, \boldsymbol{s}_{t-1}^{[m]})$
5: $\quad\quad$ $w_t^{[m]} = p(z_t \mid \boldsymbol{s}_t^{[m]})$
6: $\quad\quad$ $\overline{\mathcal{S}}_t = \overline{\mathcal{S}}_t + \left\langle \boldsymbol{s}_t^{[m]}, w_t^{[m]} \right\rangle$
7: $\quad$ **end for**
8: $\quad$ **for** $m = 1$ to M **do**
9: $\quad\quad$ draw $\boldsymbol{s}_t^{[i]}$ with probability $\propto w_t^{[i]}$
10: $\quad\quad$ $\mathcal{S}_t = \mathcal{S}_t + \left\langle \boldsymbol{s}_t^{[i]}, w_t^{[i]} \right\rangle$
11: $\quad$ **end for**
12: $\quad$ return $\mathcal{S}_t$
13: **end procedure**

update procedure is the particle set $\mathcal{S}_{t-1}$ at time $t-1$, the active control input u_t, and

the current sensor measurement z_t. It can be decomposed into the three main steps *sampling*, *weighting*, and *resampling*. Sampling is performed in line 4 of Alg. 5 creating the next particle generation $\overline{\mathcal{S}}_t$ from the prior $\mathcal{S}_{t-1}$ applying u_t. Here, $p(\boldsymbol{s}_t \mid u_t, \boldsymbol{s}_{t-1}^{[m]})$ is termed the *state transition distribution* predicting the new state at time t. Line 5 computes a weight for each particle m by evaluating $p(z_t \mid \boldsymbol{s}_t^{[m]})$ at the state $\boldsymbol{s}_t^{[m]}$. The weighting phase is also known as *importance weighting*. Resampling takes place in line 9, i.e. the samples are added to the posterior $\mathcal{S}_t$ with a probability proportional to their weights. Resampling is needed since only a finite number of particles is employed in order to approximate the target distribution. The drawback is that good particles carrying high weights might be replaced by worse particles causing a divergence of the filter. This problem is often referred to as the *particle depletion* problem in literature. However, in Monte Carlo Localization samples are drawn from the *motion model*, i.e. $p(\boldsymbol{s}_t \mid u_t, \boldsymbol{s}_{t-1}^{[i]})$ represents the probability that the new state is s_t given the previous state $\boldsymbol{s}_{t-1}^{[i]}$ and the current motion command u_t. The importance weight $w_t^{[i]}$ of particle $\boldsymbol{s}_t^{[i]}$ is computed using the *sensor model* $p(z_t \mid m, \boldsymbol{s}_t^{[i]})$ of the most recent observation z_t given the map m and the particle state $\boldsymbol{s}_t^{[i]}$. For more detailed information concerning the theory of particle filters and its peculiarities refer to Thrun *et al.*[1].

A.3. Markov Assumption

The Markov assumption or *closed world assumption* postulates that a state $\boldsymbol{x}_t$ at time t depends only on its predecessor $\boldsymbol{x}_{t-1}$, i.e.

$$p\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{0:t-1}\right) = p\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}\right) \tag{A.1}$$

This equation plays a fundamental role in mobile robot localization since Bayes filters are based on the Markov assumption. Thus, dynamic obstacles (e.g. moving people) which are not included in the state variable $\boldsymbol{x}_t$ may induce dramatic localization errors.

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. The MIT Press, 2005

Appendix B

Calculation of RPY-Angles

Given a homogenous matrix transformation $\boldsymbol{T} = \left[\begin{array}{ccc|c} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} & \boldsymbol{p} \end{array} \right]$, the orientation of the frame is calculated using Roll-Pitch-Yaw angles, i.e.

$$\boldsymbol{T}_{RPY}(\Theta_r, \Theta_p, \Theta_y) = Rot(\boldsymbol{z}, \Theta_r) Rot(\boldsymbol{y}, \Theta_p) Rot(\boldsymbol{x}, \Theta_y) \tag{B.1}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ denote the axes of the base frame.

Given $\boldsymbol{T}$, the angles Θ_r, Θ_p, and Θ_y can be computed as[1]

$$\begin{aligned} \Theta_r &= atan2(\boldsymbol{n}_y, \boldsymbol{n}_x) \\ \Theta_p &= atan2(-\boldsymbol{n}_z, cos(\Theta_r)\boldsymbol{n}_x + sin(\Theta_r)\boldsymbol{n}_y) \\ \Theta_y &= atan2(sin(\Theta_r)\boldsymbol{a}_x - cos(\Theta_r)\boldsymbol{a}_y, -sin(\Theta_r)\boldsymbol{o}_x + cos(\Theta_r)\boldsymbol{o}_y). \end{aligned} \tag{B.2}$$

[1] C. Ho and J. Sriwattanathamma. *Robot Kinematics: Symbolic Automation and Numerical Synthesis.* Ablex Pub. Corp., 1990

Appendix C

Linear Covariance Matrix Transformation

Let X be a p-dimensional random variable and let $E(X)$ and

$$\Sigma = E\left((X - E(X))(X - E(X))^T\right) \tag{C.1}$$

its mean and variance, respectively. A linear transformation

$$Y = a + BX \tag{C.2}$$

yields a q-dimensional random variable Y where a is a q-dimensional vector and B is a $(q \times p)$-Matrix. Then the transformed mean is $E(Y) = a + BE(X)$ and Σ_y is calculated as

$$\begin{aligned}
\Sigma_Y &= E\left((Y - E(Y))(Y - E(Y))^T\right) \\
&= E\left((BX - BE(X))(BX - BE(X))^T\right) \\
&= E\left(B(X - E(X))(X - E(X))^T B^T\right) \\
&= BE\left((X - E(X))(X - E(X))^T\right) B^T \\
&= B\Sigma B^T.
\end{aligned} \tag{C.3}$$

In order to understand the covariance transformation of Eqn. 2.14 let $f : \mathbb{R}^n \to \mathbb{R}^m$ consist of m real-valued functions $f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x})$ where $\boldsymbol{x} = (x_1, ..., x_n)$ is a n-dimensional random variable. The first order Taylor linearization is defined as

$$f(x_1, ..., x_n) \approx \boldsymbol{J}_f(\hat{\boldsymbol{x}}) \begin{pmatrix} x_1 - \hat{x}_1 \\ \vdots \\ x_n - \hat{x}_n \end{pmatrix} + f(\hat{x}_1, ..., \hat{x}_n). \tag{C.4}$$

In Eqn. C.4, $\hat{\boldsymbol{x}} = (\hat{x}_1, ..., \hat{x}_n)$ is the linearization point and

$$\boldsymbol{J}_f = \frac{\partial(f_1, ..., f_m)}{\partial(x_1, ..., x_n)} \tag{C.5}$$

is the Jacobian. Eqn. C.4 can be expanded to

$$f(x_1, ..., x_n) \approx \begin{pmatrix} \sum_{i=1}^{n} \frac{\partial f_1(\hat{\boldsymbol{x}})}{\partial x_i} x_i \\ \vdots \\ \sum_{i=1}^{n} \frac{\partial f_m(\hat{\boldsymbol{x}})}{\partial x_i} x_i \end{pmatrix} + f(\hat{\boldsymbol{x}}) + \boldsymbol{J}_f(\hat{\boldsymbol{x}})\hat{\boldsymbol{x}} \tag{C.6}$$

Eqn. C.6 implies that each random variable x_i is linearly transformed. Consequently, given the covariance matrix $\boldsymbol{P}_j$ of a k-dimensional sub-vector $\boldsymbol{x}_k = (x_i, .., x_{i+k})$ of $\boldsymbol{x}$, the transformed covariance matrix is calculated as

$$\boldsymbol{P}'_j = \boldsymbol{J}_{f,x_k} \boldsymbol{P}_j \boldsymbol{J}_{f,x_k}^T \tag{C.7}$$

with

$$\boldsymbol{J}_{f,x_k} = \frac{\partial(f_1, ..., f_m)}{\partial \boldsymbol{x}_k} \tag{C.8}$$

Since f involves a linear combination of its random variables, the covariance matrix $\boldsymbol{P}_f$ is calculated as the sum of the transformed covariance matrices of its individual elements:

$$\boldsymbol{P}_f = \sum_{j=1}^{l} \boldsymbol{P}'_j \tag{C.9}$$

For example, consider the covariance transformation shown in Eqn. 2.14. The input vector of the function $\boldsymbol{h}_{i,j} = h\left(\hat{\boldsymbol{x}}_B^A, \boldsymbol{p}_i, \boldsymbol{q}_j\right)$ can be expanded to $\boldsymbol{x} = (x_x, x_y, y_\Theta, p_x, p_y, q_x, q_y)$. Thus, $\boldsymbol{x}$ comprises three sub-vectors $\hat{\boldsymbol{x}}_B^A$, $\boldsymbol{p}_i$, and $\boldsymbol{q}_j$. As a result, three covariance matrices are added in Eqn. 2.14.

Appendix D

Derivation of ∇e

According to equation 5.26, the following mean square error function has to be minimized:

$$
\begin{aligned}
e\left(\boldsymbol{\delta}\right) &= \sum_{i=N_1}^{N_n} \left[\boldsymbol{H}_{best}{}^{\mathcal{LM}}\boldsymbol{T}_{f_N}\boldsymbol{T}_{\delta x,\delta y,\delta\phi}\boldsymbol{x}_i^{\mathcal{LM}} - \boldsymbol{x}_i^{\mathcal{GM}}\right]^2 \\
&= \sum_{i=N_1}^{N_n} \left[\underbrace{\begin{pmatrix} c\beta & -s\beta & t_x^{H,f} \\ s\beta & c\beta & t_y^{H,f} \\ 0 & 0 & 1 \end{pmatrix}}_{=\boldsymbol{H}_{best}{}^{\mathcal{LM}}\boldsymbol{T}_{f_N}} \underbrace{\begin{pmatrix} c\delta\phi & -s\delta\phi & \delta x \\ s\delta\phi & c\delta\phi & \delta y \\ 0 & 0 & 1 \end{pmatrix}}_{=\boldsymbol{T}_{\delta x,\delta y,\delta\phi}} \begin{pmatrix} x_i^{\mathcal{L}} \\ y_i^{\mathcal{L}} \\ 1 \end{pmatrix} - \begin{pmatrix} x_i^{\mathcal{G}} \\ y_i^{\mathcal{G}} \\ 1 \end{pmatrix} \right]^2 \\
&= \sum_{i=N_1}^{N_n} \left[\begin{pmatrix} p_x^{\mathcal{G}}(\delta x,\delta y,\delta\phi) \\ p_y^{\mathcal{G}}(\delta x,\delta y,\delta\phi) \\ 1 \end{pmatrix} - \begin{pmatrix} x_i^{\mathcal{G}} \\ y_i^{\mathcal{G}} \\ 1 \end{pmatrix} \right]^2
\end{aligned}
\tag{D.1}
$$

The letters s and c abbreviate sin and cos. The angle β is calculated by appling the $atan2$ function on $\boldsymbol{H}_{best}{}^{\mathcal{LM}}\boldsymbol{T}_{f_N}$. Note that 3×3 matrices are used since it is assumed that the robot moves on a plane. The gradient is calculated as

$$
\begin{aligned}
&\nabla \sum_{i=N_1}^{N_n} \left[\begin{pmatrix} p_x^{\mathcal{G}}(\delta x, \delta y, \delta\phi) \\ p_y^{\mathcal{G}}(\delta x, \delta y, \delta\phi) \\ 1 \end{pmatrix} - \begin{pmatrix} x_i^{\mathcal{G}} \\ y_i^{\mathcal{G}} \\ 1 \end{pmatrix} \right]^2 \\
&= \sum_{i=N_1}^{N_n} \nabla \left(\left(p_x^{\mathcal{G}}(\delta x, \delta y, \delta\phi) - x_i^{\mathcal{G}} \right)^2 + \left(p_y^{\mathcal{G}}(\delta x, \delta y, \delta\phi) - y_i^{\mathcal{G}} \right)^2 \right) \\
&= \sum_{i=N_1}^{N_n} \begin{pmatrix} 2\left(p_x^{\mathcal{G}} - x_i^{\mathcal{G}}\right) \frac{\partial}{\partial \delta x} p_x^{\mathcal{G}} + 2\left(p_y^{\mathcal{G}} - y_i^{\mathcal{G}}\right) \frac{\partial}{\partial \delta x} p_y^{\mathcal{G}} \\ 2\left(p_x^{\mathcal{G}} - x_i^{\mathcal{G}}\right) \frac{\partial}{\partial \delta y} p_x^{\mathcal{G}} + 2\left(p_y^{\mathcal{G}} - y_i^{\mathcal{G}}\right) \frac{\partial}{\partial \delta y} p_y^{\mathcal{G}} \\ 2\left(p_x^{\mathcal{G}} - x_i^{\mathcal{G}}\right) \frac{\partial}{\partial \delta \phi} p_x^{\mathcal{G}} + 2\left(p_y^{\mathcal{G}} - y_i^{\mathcal{G}}\right) \frac{\partial}{\partial \delta \phi} p_y^{\mathcal{G}} \end{pmatrix}
\end{aligned}
\tag{D.2}
$$

with

$$
\begin{aligned}
\frac{\partial}{\partial \delta x} p_x^{\mathcal{G}} &= c\beta \\
\frac{\partial}{\partial \delta x} p_y^{\mathcal{G}} &= s\beta \\
\frac{\partial}{\partial \delta y} p_x^{\mathcal{G}} &= -s\beta \\
\frac{\partial}{\partial \delta y} p_y^{\mathcal{G}} &= c\beta \\
\frac{\partial}{\partial \delta \phi} p_x^{\mathcal{G}} &= y_i^{\mathcal{L}} \left(s\beta s\delta\phi - c\beta c\delta\phi \right) - x_i^{\mathcal{L}} \left(c\beta s\delta\phi + s\beta c\delta\phi \right) \\
\frac{\partial}{\partial \delta \phi} p_y^{\mathcal{G}} &= x_i^{\mathcal{L}} \left(c\beta c\delta\phi - s\beta s\delta\phi \right) - y_i^{\mathcal{L}} \left(sc\beta c\delta\phi + c\beta s\delta\phi \right)
\end{aligned}
\tag{D.3}
$$

Appendix E

Mathematical Derivation of $\mathcal{W}(\mathcal{H})$

In order to provide a justification of the target function defined in Eqn. 3.13, the state variable representing the SLAM problem is formulated in a recursive manner. The following steps are based on the mathematical derivation of the GraphSLAM algorithm described by Thrun *et al.*[1]. Further on, $\boldsymbol{y}_t$ denotes a state variable which combine the map m, the robot pose $\boldsymbol{x}_t^r$ at time t, and the origin of the fragment $\boldsymbol{x}_t^f$ with respect to its predecessor $\boldsymbol{x}_{t-1}^f$:

$$\boldsymbol{y}_{0:t} = \begin{pmatrix} \boldsymbol{x}_0^f \\ \boldsymbol{x}_1^f \\ \vdots \\ \boldsymbol{x}_t^f \\ m \\ \boldsymbol{x}_t^r \end{pmatrix} \text{ and } \boldsymbol{y}_t = \begin{pmatrix} \boldsymbol{x}_t^f \\ m \\ \boldsymbol{x}_t^r \end{pmatrix} \tag{E.1}$$

A discrete timestep from $t-1$ to t indicates the initialization of a new map fragment. As a result, the *length* of the timesteps in terms of robot motions are not necessarily equidistant. The map variable m composes all fragment points. The vector $\boldsymbol{y}_{0:t}$ represents the complete history of all generated map fragments. The full SLAM posterior is defined as $p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right)$. The variable z_t denotes an observation or *measurement*. In this context, an observation composes loop closing investigations. If successful, z_t represents the map matching result $h(a,i)$ (cf. p. 56). Finally, $\boldsymbol{\tau}_{1:t}$ provides relative transformations between adjacent fragment. According to Bayes rule the posterior may be factored as

$$p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right) = \eta p\left(z_t \mid \boldsymbol{y}_{0:t}, z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) p\left(\boldsymbol{y}_{0:t} \mid z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right). \tag{E.2}$$

Unnecessary variables of the first factor on the right hand side can safely be discarded:

[1] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. The MIT Press, 2005

$$p\left(z_t \mid \boldsymbol{y}_{0:t}, z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) = p\left(z_t \mid \boldsymbol{y}_t\right) \tag{E.3}$$

The second factor can be factorized by partitioning $\boldsymbol{y}_{0:t}$ into $\boldsymbol{x}_t^f, m, \boldsymbol{x}_t^r$ and $\boldsymbol{y}_{0:t-1}$. Unnecessary conditioning variables are dropped again.

$$\begin{aligned}
&p\left(\boldsymbol{y}_{0:t} \mid z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) \\
&= p\left(\boldsymbol{x}_t^f, m, \boldsymbol{x}_t^r \mid \boldsymbol{y}_{0:t-1}, z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) p\left(\boldsymbol{y}_{0:t-1} \mid z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) \\
&= p\left(\boldsymbol{x}_t^f \mid \boldsymbol{y}_{0:t-1}, z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) p\left(\boldsymbol{y}_{0:t-1} \mid z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) \\
&= p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right) p\left(\boldsymbol{y}_{0:t-1} \mid z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right)
\end{aligned} \tag{E.4}$$

Putting Eqns. E.3 and E.4 back into Eqn. E.2 yields the following recursive update equation:

$$p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right) = \eta p\left(z_t \mid \boldsymbol{y}_t\right) p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right) p\left(\boldsymbol{y}_{0:t-1} \mid z_{1:t-1}, \boldsymbol{\tau}_{1:t}\right) \tag{E.5}$$

Finally, induction over t facilitates to state a closed form expression of the posterior[1]. $p(\boldsymbol{y}_0)$ is the prior knowledge about the map.

$$\begin{aligned}
p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right) &= \eta p(\boldsymbol{y}_0) \prod_t p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right) p\left(z_t \mid \boldsymbol{y}_t\right) \\
&= \eta' \prod_t p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right) p\left(z_t \mid \boldsymbol{y}_t\right)
\end{aligned} \tag{E.6}$$

The latter equation holds under the assumption that no map prior is given. In order to derive the target function 3.13, Bayes rule is applied again, i.e.

$$\begin{aligned}
p\left(z_t \mid \boldsymbol{y}_t\right) &= \eta p\left(\boldsymbol{y}_t \mid z_t\right) \underbrace{p\left(z_t\right)}_{\approx const.} \\
&\propto p\left(\boldsymbol{y}_t \mid z_t\right).
\end{aligned} \tag{E.7}$$

Thus,

$$p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right) = \eta'' \prod_t p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right) p\left(\boldsymbol{y}_t \mid z_t\right). \tag{E.8}$$

Computing the negative log-likelihood yields

$$-log\left(p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right)\right) = -log(\eta) - \left(\sum_t log\left(p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right)\right) + log\left(p\left(\boldsymbol{y}_t \mid z_t\right)\right)\right) \tag{E.9}$$

The inter-fragment transformations and the measurement process are assumed to be normal distributed. Consequently,

$$\begin{aligned} p\left(\boldsymbol{x}_t^f \mid \boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t\right) &= \eta e^{-\frac{1}{2}\left(\boldsymbol{x}_t^f - g(\boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t)\right)\left(\boldsymbol{\Sigma}_{(t-1)t}^f\right)^{-1}\left(\boldsymbol{x}_t^f - g(\boldsymbol{x}_{t-1}^f, \boldsymbol{\tau}_t)\right)^T} \\ &= \eta e^{-\frac{1}{2}\left(\boldsymbol{x}_t^f - \boldsymbol{\mu}_{(t-1)t}\right)\left(\boldsymbol{\Sigma}_{(t-1)t}^f\right)^{-1}\left(\boldsymbol{x}_t^f - \boldsymbol{\mu}_{(t-1)t}\right)^T} \end{aligned} \tag{E.10}$$

and

$$\begin{aligned} p\left(\boldsymbol{y}_t \mid z_t\right) &= \eta e^{-\frac{1}{2}(\boldsymbol{x}_t^r - u(z_t))\boldsymbol{\Sigma}_{h(f_t, f_{t'})}^{-1}(\boldsymbol{x}_t^r - u(z_t))^T} \\ &= \eta e^{-\frac{1}{2}\left(\boldsymbol{x}_t^r - \boldsymbol{\mu}_{h(f_t, f_{t'})}\right)\boldsymbol{\Sigma}_{h(f_t, f_{t'})}^{-1}\left(\boldsymbol{x}_t^r - \boldsymbol{\mu}_{h(f_t, f_{t'})}\right)^T} \end{aligned} \tag{E.11}$$

Here, $g(\cdot)$ is a function applying $\boldsymbol{\tau}_t$ to $\boldsymbol{x}_{t-1}^f$ and $u(\cdot)$ is a measurement function which projects $\boldsymbol{x}_t^r$ to the revisited fragment $f_{t'}$ by matching f_t to $f_{t'}$. The result is an expectation value of the correct robot pose denoted by $\boldsymbol{\mu}_{h(f_t, f_{t'})}$ (cf. p. 56). Replacing the summands of Eqn. E.9 by E.10 and E.11 yields

$$\begin{aligned} & - log\left(p\left(\boldsymbol{y}_{0:t} \mid z_{1:t}, \boldsymbol{\tau}_{1:t}\right)\right) \\ &= -log(\eta'') + \frac{1}{2}\sum_t \left(\boldsymbol{x}_t^f - \boldsymbol{\mu}_{(t-1)t}\right)\left(\boldsymbol{\Sigma}_{(t-1)t}^f\right)^{-1}\left(\boldsymbol{x}_t^f - \boldsymbol{\mu}_{(t-1)t}\right)^T \\ &\quad + \left(\boldsymbol{x}_t^r - \boldsymbol{\mu}_{h(f_t, f_{t'})}\right)\boldsymbol{\Sigma}_{h(f_t, f_{t'})}^{-1}\left(\boldsymbol{x}_t^r - \boldsymbol{\mu}_{h(f_t, f_{t'})}\right)^T \end{aligned} \tag{E.12}$$

which reflects the structure of $\mathcal{W}(\mathcal{H})$ (cf. Eqn. 3.13).

Appendix F

Own Publications

- R. Iser and F. Wahl. AntSLAM: Global map optimization using swarm intelligence. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 265 – 272, 2010
- R. Iser, A. Martens, and F. Wahl. Localization of mobile robots using incremental local maps. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 4873 – 4880, 2010
- U. Thomas and R. Iser. A new probabilistic path planning algorithm for (dis)assembly tasks. In *Proc. of ISR/Robotik*, pages 467 – 472, 2010
- D. Kubus, R. Iser, S. Winkelbach, and F. Wahl. Efficient parallel random sample matching for estimation, localization, and related problems. In *German Workshop on Robotics*, pages 239 – 250, 2009
- R. Iser, D. Kubus, and F. Wahl. An efficient parallel approach to random sample matching (pRANSAM). In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1199 – 1206, 2009
- R. Iser and F. Wahl. Building local metrical and global topological maps using efficient scan matching approaches. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 1023 – 1030, 2008
- R. Iser, J. Spehr, S. Winkelbach, and F. Wahl. Mobile robot localization using the fast random sample matching approach. In *Proc. of ISR/Robotik*, pages 163 – 166, 2008
- U. Thomas, S. Molkenstruck, R. Iser, and F. Wahl. Multi sensor fusion in robot assembly using particle filters. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3837 – 3843, 2007

List of Figures

List of Tables

Index